Graph manipulation for graph-based simultaneous localization and mapping

Askhat Issakov

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.
Espoo 22.02.2021

Supervisor

Prof. Ville Kyrki

Advisor

D.Sc. Juhana Ahtiainen

Aalto University
School of Electrical Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Askhat Issakov

Title Graph manipulation for graph-based simultaneous localization and mapping

Degree programme ICT Innovation

Major Autonomous Systems **Code of major** ELEC3055

Supervisor Prof. Ville Kyrki

Advisor D.Sc. Juhana Ahtiainen

Date 22.02.2021 **Number of pages** 65+9 **Language** English

Abstract

Simultaneous localization and mapping (SLAM) is a fundamental problem in robotic navigation. It consists of two tasks: building a map of the environment and localizing a robot within it. Graph SLAM accomplishes these tasks by constructing and optimizing a graph of relations. Most solutions for SLAM work with default parameters in most environments but fail in challenging environments. For example, indoor spaces with no satellite navigation (GNSS) signal and a lot of repetitive features (patterns) make SLAM fail to build a consistent graph in environments such as supermarkets and warehouses. However, human experts can help to correct the resulting graph inconsistencies given enough visual information about the environment.

The aim of this thesis is to develop a software application for correcting algorithmic SLAM failures by allowing human experts to edit the underlying graph. The system was developed using standard system engineering practices. The proposed Graph Manipulation Application allows the user to correct a 3D map of the environment built by the automatic SLAM process. The user can manipulate graph nodes and edges through the graphical user interface, register local submaps, optimize the global map estimate, load and save existing maps.

The solution was evaluated using two challenging data sets: indoor warehouse space with repetitive shelves and outdoor urban route with tall buildings. The evaluation results demonstrate that the proposed system improved the consistency of the global map estimates in all conducted experiments. Additionally, this thesis developed a basic framework for evaluation of mapping results and basic workflows for correcting indoor and outdoor maps. However, future work is needed to extend the framework and verify the results on more data sets.

Keywords Simultaneous localization and mapping (SLAM), pose graph SLAM, human-in-the-loop, mobile robotics

Preface

This work was done at GIM Robotics with the support and help of all the amazing people there. I am really grateful that I have the opportunity to learn from my colleagues who are at the forefront of robotics development and research in Finland and beyond. First and foremost, I would like to thank my advisor D.Sc. Juhana Ahtiainen for his endless patience, tremendous help and valuable insights. Thanks to my teammates and especially Janne Paanajärvi, Petri Karppinen and Ilkka Oksanen for fruitful discussions, reviews and ideas. Thanks to everyone else at GIM Robotics who helped me in myriad smaller ways.

I would like to thank my supervisor Prof. Ville Kyrki for the careful review and guidance of this work, that helped me make it better.

Finally, this work would not have been possible without the unwavering support and love of my family, my wife Laura and my daughter Alima. They kept me going in the most difficult times.

Espoo, 22.02.2021 Askhat Issakov

Contents

Symbols and abbreviations

Symbols and operators

x	A scalar
$\mathbf{x}$	A vector
$\mathbf{X}$	A matrix
X	A set of individual values
$\mathbf{x}^T$	The transpose of $\mathbf{x}$
$\mathbf{x}^{-1}$	The inverse of $\mathbf{x}$
X_T	The set $\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_T$
$\{a, b, c\}$	A set of individual elements
$P(\mathbf{x})$	The probability distribution over $\mathbf{x}$
$P(\mathbf{x}\|\mathbf{z})$	The conditional probability distribution over $\mathbf{x}$, conditioned on $\mathbf{z}$
e^x	The exponent function of x
$\log(x)$	The logarithmic function of x
$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\mathbf{x}$ is normally distributed with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
$\sigma_{\mathbf{x}}$	The standard deviation of $\mathbf{x}$
$\sum_{i=1}^{n}$	The sum of elements over index i from 1 to n

Abbreviations

2D	Two-dimensional
3D	Three-dimensional
API	Application Programming Interface
APRIL	Autonomy, Perception, Robotics, Interfaces, and Learning
C4	Context, Containers, Components, and Code
D2D	Distribution-To-Distribution
DBN	Dynamic Bayesian Network
g^2o	General Graph Optimization
GIM	Generic Intelligent Machines
GNSS	Global Navigation Satellite System
GTSAM	Georgia Tech Smoothing and Mapping
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
ICP	Iterative Closest Point
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
iSAM	Incremental Smoothing and Mapping
ISO	International Standard Organization
JSON	JavaScript Object Notation
LiDAR	Light Detection and Ranging
MAGIC	Multi Autonomous Ground-robotic International Challenge
MRF	Markov Random Field
NASA	National Aeronautics and Space Administration
NDT	Normal Distribution Transform
OM	Occupancy Map
P2D	Point-To-Distribution
PCD	Point Cloud Data
RGB-D	Red, Green, Blue, Depth
ROS	Robot Operating System
RTK	Real-Time Kinematic
SE	Systems Engineering
SLAM	Simultaneous Localization and Mapping
SPA	Sparse Pose Adjustment
WebGL	Web Graphics Library

1 Introduction

Steady growth in commercial applications of mobile robots [1] and increased performance requirements for automated vehicles [2] has renewed interest in the old problem of simultaneous localization and mapping known as simply SLAM. The core of the problem is an estimation of the robot's location within a map of the environment that is being constructed at the same time. The estimation is highly uncertain because both location and map are unknown beforehand and the only input a robot receives is the noisy on-board sensor measurements. Decades of successful theoretical research and practical implementations make a convincing case for declaring SLAM solved [3]. However, the success is highly dependent on robot configuration and motion, environment type and performance requirements [4]. As a result, SLAM algorithms may still fail silently for some combinations of these variables and produce non-usable maps.

Modern SLAM systems consist of two main parts, front-end and back-end, and an interface between them, graph-based map representation [4]. Figure 1 illustrates the complete SLAM pipeline. The front-end performs sensor data processing and association while the back-end searches for the best map configuration based on the processed sensor measurements provided by the front-end. The interface between them is usually represented as a graph of robot poses and its constraints. This adds a task of building a graph-based map representation to the front-end.

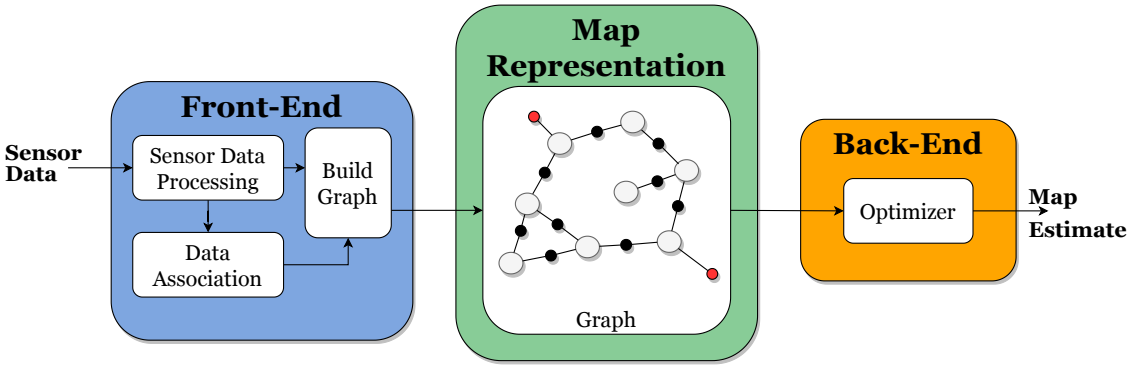

Figure 1: SLAM pipeline consists of the front-end, the back-end and the graph-based map representation interface between them. Detailed description of the SLAM pipeline is presented in Section 2.2. Figure adapted from [5].

According to Cadena et al. [4], there are three main types of SLAM failures: hardware-related, software-related and algorithmic failures. Hardware-related SLAM failures occur because of sensor or actuator defects or degradation. Software-related SLAM failures are mostly the result of poor software development practices including software testing. Algorithmic SLAM failures are the hardest to detect and mitigate because they are usually implicit in the incorrect graph structure passed between the front-end and the back-end. This means that the back-end assumes that the graph structure it receives from the front-end is always correct. This assumption works pretty well for short-term data association when the consecutive measurements are frequent and the front-end has no problems in recognizing and associating map

features with small changes. However, it is impossible to guarantee for long-term data association where the front-end has to compare, recognize and associate map features with large spatial and temporal changes. The most common problem is the phenomenon of perceptual aliasing when similar-looking places, as perceived by the robot sensors, make the front-end to associate non-related map features (false positives). On the other hand, when the front-end fails to associate related measurements (false negatives), it reduces the back-end estimation accuracy because of fewer relative constraints used in the estimation. This thesis focuses on algorithmic SLAM failures, which are discussed in more detail in Section 2.3.

The aim of this thesis is to develop a software application for mitigating algorithmic SLAM failures with the help of a human expert by editing the underlying graph. This allows to leverage the strengths of both the robot and the human. The human is good at seeing the high-level picture and integrating the contextual information while the robot is good at precise calculations needed for map registration and optimization. Therefore, the main research problem of this thesis is:

How human expertise can be used to mitigate algorithmic SLAM failures in different challenging environments?

This thesis divides the main problem into three research questions:

RQ1 : *What are the main requirements for the design and development of a graph manipulation application?*
RQ2 : *Is the proposed solution effective in fixing a broken map?*
RQ3 : *What are the most effective workflows (sequences of actions) for fixing indoor and outdoor maps?*

Figure 2 illustrates a modified SLAM pipeline proposed in this thesis. Proprietary SLAM software stack of GIM (Generic Intelligent Machines) Robotics is used as the base SLAM pipeline on top of which the proposed software application is developed. Therefore, some design choices are influenced by the existing software stack. More details about requirements and design choices are presented in Chapters 3 and 4.

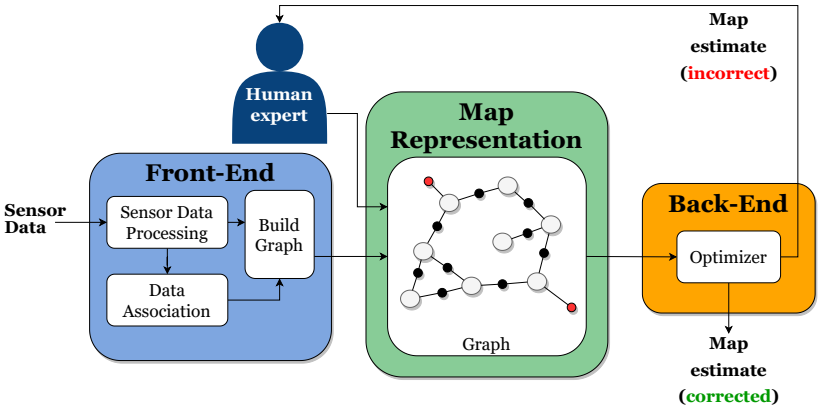

Figure 2: Proposed SLAM pipeline including the human expert.

This thesis is organized as follows. Chapter 2 sets the context and describes motivation for the current work by reviewing prior research on SLAM and its failures as well as existing solutions. Chapter 3 presents the requirements analysis that affected the system design choices. This chapter answers the research question RQ1 using standard methodologies to define the main stakeholder needs and system requirements. Chapter 4 details the system design and its main components, which address previously defined system requirements. Chapter 5 lists experiments performed with the system. This chapter addresses the remaining research questions RQ2 and RQ3 by defining evaluation methodology and comparing results of user manipulations on outdoor and indoor environment data sets. Chapter 6 concludes this thesis and outlines future work.

2 Background

The problem of SLAM is one of the most fundamental and extensively researched problems in robotics. It was first discussed at the 1986 IEEE Robotics and Automation Conference and later formulated in key papers by a group of researchers that included Peter Cheeseman, Jim Crowley, and Hugh Durrant-Whyte [6]. Since then there have been three distinct periods of research in SLAM according to Cadena et al. [4]. The first period from 1986 until 2004 laid foundations by introducing the main probabilistic formulations for the SLAM problem [6]–[8]. The second period between 2004 and 2015 focused on fundamental properties of SLAM, the key role of sparsity for efficient SLAM solvers and open-source SLAM implementations [9]. The ongoing period of SLAM research addresses the issues of robust performance, high-level environment understanding, robot resource awareness and task-driven perception [4]. This thesis addresses the issue of robustness in map construction in line with the current focus of academia.

This chapter sets the background and context for the thesis work. First, Section 2.1 gives a short overview of different SLAM formulations. Next, Section 2.2 describes a generic SLAM pipeline and its components. Finally, Section 2.3 zooms in on algorithmic SLAM failures, challenges in the environment and overviews existing solutions addressing them.

2.1 SLAM overview

There are two main types of SLAM problem formulations: full and online. The online SLAM jointly estimates *the current robot pose* and the map given the history of sensor measurements until the estimation time. While the full SLAM jointly estimates *the full robot trajectory* and the map given the full history of sensor measurements. To make the definitions more explicit let us use the mathematical notation and start with the online SLAM.

We denote the current robot pose as $\mathbf{x}_t$ and the map as M. Notice that the map does not depend on time since we assume it to be static[1], i.e. does not change over time. As for the sensor measurements, we distinguish two types of sensors: motion (or odometry) sensors that keep track of the robot's movement and environment sensors that perceive the surrounding environment. The first group of sensors includes wheel encoders, gyroscopes and IMU among others. The second group includes laser range finders (LiDAR), cameras (mono, stereo, RGB-D) and radars among others. We define odometry sensor measurements as U_{t-1} until time $t-1$ and environment sensor measurements as Z_t until time t. Now we can put everything together and define the online SLAM probabilistically:

$$P(\mathbf{x}_t, M | U_{t-1}, Z_t). \tag{1}$$

Note that the odometry measurements are given until $t-1$ time, i.e. one less time step than the environment measurements at time t. This is because odometry

[1] Another possibility is that the map is dynamic, i.e. environment features change over time. Some of the solutions are presented in Section 2.3.1.

measurements are dependent on robot motion and relate two consecutive robot poses while environment sensors perceive the environment continuously. Put it another way, odometry measurements are also called *control input* because they predict the robot's target pose based on control inputs at the source pose. Then taking a snapshot at time t means that the latest available odometry measurement (prediction) was recorded at time $t-1$ and relates poses $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$.

For the full SLAM, we denote the full robot trajectory as $X_T = \{\mathbf{x}_0, ..., \mathbf{x}_T\}$ and the map as M given the full history of odometry U_T and other sensor Z_T measurements, which gives us the following equation:

$$P(X_T, M|U_T, Z_T). \tag{2}$$

A nonlinear *motion model* function f is usually used to describe the relationship between two consecutive poses and the corresponding odometry measurement:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \mathbf{w}_t, \tag{3}$$

where the noise $\mathbf{w}_t$ is modelled as zero-mean Gaussian with covariance $\mathbf{\Sigma}_t$. Therefore, Equation 3 is modelled as a Gaussian distribution as well:

$$\mathbf{x}_{t+1} \sim \mathcal{N}(f(\mathbf{x}_t, \mathbf{u}_t), \mathbf{\Sigma}_t).$$

The corresponding *sensor model* function h helps to predict the sensor measurements given the current robot pose and map:

$$\mathbf{z}_t = h(\mathbf{x}_t, M) + \lambda_t, \tag{4}$$

where λ_t models the sensor noise as zero-mean Gaussian with covariance $\mathbf{\Lambda}_t$. Therefore, Equation 4 is modelled as a Gaussian distribution as well:

$$\mathbf{z}_t \sim \mathcal{N}(h(\mathbf{x}_t, M), \mathbf{\Lambda}_t).$$

Using the motion model in Equation 3 we can think of odometry sensors as internal (proprioceptive) sensors that do not depend on the external map. On the other hand, environment sensors are external (exteroceptive) sensors that depend on the environment based on the sensor model (Equation 4).

Originally, the online SLAM formulation was considered more suitable for online estimation (run both mapping and localization simultaneously) while the full SLAM for offline estimation (run mapping first to build a map and then run localization to localize within the ready map). However, more recent work [10] showed that these two formulations can be merged in incremental smoothing approaches, where SLAM runs online while estimating the full trajectory until the current time t.

$$P(X_t, M|U_{t-1}, Z_t). \tag{5}$$

While the ultimate goal of SLAM research is to have a robot that can successfully navigate in any previously unseen environment online, the reality of challenging man-made environments makes it difficult to achieve online operations robustly and

guarantee safety. Therefore, it seems reasonable to perform offline estimation and break down SLAM estimation into distinct mapping and localization stages. First, the robot maps the environment offline. Second, the robot moves and localizes itself in the mapped area online. This is called map-based navigation. This thesis focuses on the mapping stage and correctness of the map estimate produced by the graph-based SLAM pipeline.

2.2 Graph-based SLAM pipeline

Modern SLAM solutions divide the task of building a map between the front-end and the back-end with a map representation interface between them (Figure 1). Section 2.2.1 describes how the front-end processes the incoming sensor measurements to build an abstract map representation for the back-end. Next, Section 2.2.3 details the process of map optimization within the back-end. Finally, Section 2.2.2 explains the factor graph and its components.

2.2.1 Front-end

The main task of the SLAM front-end is to transform incoming raw sensor measurements into a graph representation that can be used by the SLAM back-end to find the optimal graph topology. This is achieved in several stages. First, the front-end has to preprocess the incoming raw sensor measurements into a map representation suitable for data association. Next, preprocessed sensor measurements have to be associated with each other to find correct matches. Finally, all constraints have to be combined into a graph structure. The whole front-end workflow is shown on Figure 3.

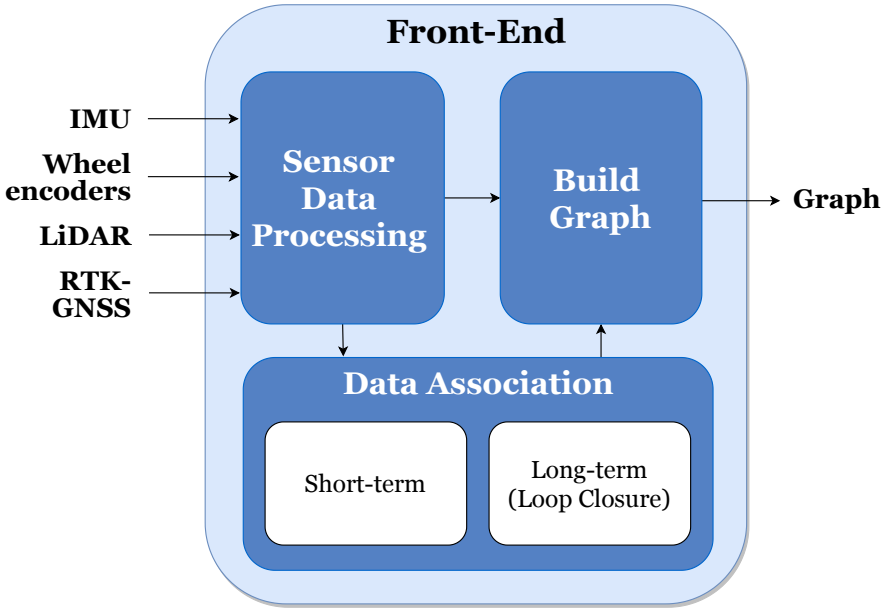

Figure 3: SLAM front-end consists of three main components: sensor data processing, data association and graph building. Data association is further broken down into short-term and long-term scan matching.

Modern robotic systems use several sensors to perceive the environment. This allows for combining sensors with complimentary features to achieve a more complete picture of the surroundings. Another benefit is that using redundant sensors makes the whole system more robust to hardware failures and provides fallback options for such cases. While proprioceptive sensors such as IMU and wheel encoders provide very important pieces of information about the robot's location and pose, most mobile robots use lasers and cameras as primary sensors to perceive and model the surrounding environment. The choice of the primary exteroceptive sensor used for perception defines the map representation used for data association. Therefore, the SLAM front-end depends on the choice of the primary sensor. Both camera-based and laser-based front-ends have their benefits and drawbacks [11], which are summarised in Table 1.

Table 1: Camera-based vs laser-based SLAM front-ends

Feature	**Camera**	**LiDAR**
Type	Passive	Active
Range	Long	Medium
Precision and accuracy	Medium	High
Cost	Low to Medium	Medium to High
Working in dark	No	Yes
Includes distance and location data	No	Yes
Can detect road features	Yes	Limited
Output	image	point cloud

Sensor data processing

The first stage in the SLAM front-end workflow is sensor data processing which transforms the raw sensor measurements into a map representation suitable for data association. Traditionally, there are three broad types of map representation (Figure 4). Grid-based maps represent the environment as the grid of occupied and unoccupied cells and they are usually associated with laser-based front-ends, which allow for easier space discretization because they provide distance data. Feature-based maps represent the environment as the map of the prominent landmark or feature locations and favor camera-based front-ends. Topological maps represent the environment as the graph of nodes and their connections.

There are many specific map representations within these categories and some of the most prominent ones are:

- Occupancy Grid [13]
- Octomap [14]
- Normal Distributions Transform (NDT) [15]
- 3D Normal Distributions Transform Occupancy Map (NDT-OM) [16]
- Pose Graph [17]
- Factor Graph [18]

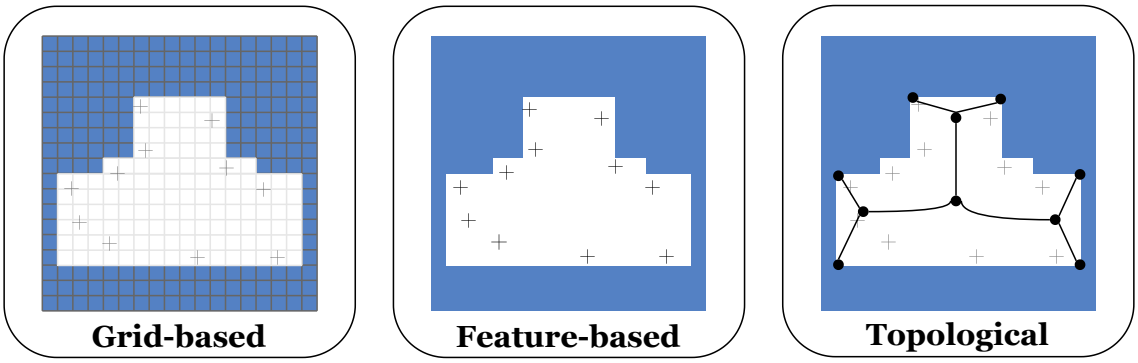

Figure 4: Map representations. Grid-based maps are usually used by laser-based SLAM front-ends, feature-based maps favor camera-based SLAM front-ends and topological maps provide a more compact representation of the environment. Figure adapted from [12].

This thesis uses NDT-OM and pose graph as the corresponding map representations for data association and map optimization since those are the map formats used in GIM SLAM software.

Data association

The second stage of data association in the SLAM front-end is a critical stage because its task is to associate incoming sensor measurements with previous measurements. Simply put, it has to recognize that different measurements see in fact the same place. This task is further categorized into short-term and long-term data association. The former associates consecutive environment scans with only small changes while the latter associates scans with large spatial and temporal changes. Long-term data association is also called a loop closure following the analogy of returning to the previously seen place and therefore making a loop. While usually, the underlying scan matching algorithm is the same, short-term and long-term data association are quite different in terms of reliability. As already mentioned in the introduction, loop closure is challenging because of perceptual aliasing and often times produces false positive associations.

Some of the most widely used scan matching algorithms are the following:

- NDT map representation based algorithms (NDT [15], D2D-NDT [19], P2D-NDT [20])
- Point cloud map representation based algorithms (ICP [21])

This thesis uses the D2D-NDT algorithm for scan matching (map registration) implemented in GIM SLAM software.

Graph construction

Finally, the SLAM front-end combines all graph constraints and nodes into a complete graph structure that could be passed to the back-end. There are several open-source back-end libraries available that could be used to construct a graph: g^{2}o [22], iSAM [23], GTSAM [24]. The general procedure for graph construction is the following:

1. Add an initial node

2. Add absolute constraints on the initial node
3. Add the rest of nodes
4. Add the rest of absolute constraints
5. Add all relative constraints between the nodes

This thesis uses iSAM library to construct and optimize the graph. The next section describes graph-based map representations most often used for map optimization.

2.2.2 Graph-based map representation

Representing environment as grid-based or feature-based maps has advantages of capturing the external world well but one downside is the large size of such maps. Topological or graph maps provide a more compact map representation and they are also well suited for the optimization formulation [17]. Figure 5 shows different graph-based map representations used primarily for map optimization by the SLAM back-end.

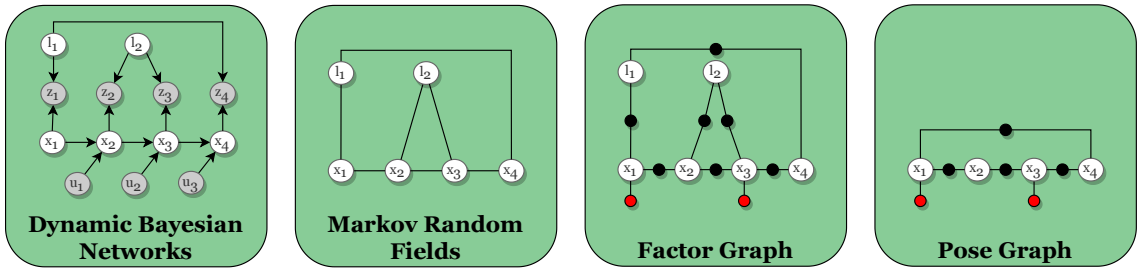

Figure 5: Graph-based map representations are used by the SLAM back-end for map optimization. Figure adapted from [5].

Dynamic Bayesian Networks (DBN) [25] are directed acyclic graphs that represent observed (odometry and landmark measurements) and unobserved (robot poses and landmark positions) variables as nodes and dependencies between them as edges. Figure 5 illustrates observed variables as white nodes and unobserved variables as gray nodes. More details about DBNs for SLAM and state estimation can be found in [26]. Markov Random Fields (MRF) [27] are undirected graphs that represent robot poses and landmark positions as nodes and sensor measurements as edges and allow for cyclic connections as well.

Factor graphs [18] are bipartite undirected graphs that represent both variables (robot poses and landmark positions) and relations between them (factors) as nodes and their dependencies as edges. Figure 5 shows variables as white nodes, bifactors as black nodes and unifactors as red nodes. Finally, pose graphs [17] is a more general graph-based map representation that represents only robot poses as nodes and applies to all three previous graph types.

2.2.3 Back-end

The SLAM back-end uses the graph provided by the front-end to solve an optimization problem. The map optimization problem could be formulated using such algorithms as nonlinear least squares [10], [28]–[30], relaxation methods [31]–[33], linear methods [34], [35], stochastic gradient descent [36], [37]. The back-end tries to find the optimal graph topology based on constraints passed in by the front-end. Since the front-end has already preprocessed raw sensor measurements into the graph-based map representation, the SLAM back-end is sensor-agnostic, which means it does not depend on sensors used to perceive the environment. This makes the whole pipeline flexible because it allows to use different back-ends based on their performance independent from the front-end. iSAM library, which is used as the back-end optimizer in this thesis, provides a general non-linear least squares method for optimizing factor graphs.

2.3 Algorithmic SLAM failures

As was already mentioned in the introduction, the source of the incorrect map estimates produced by the graph-based SLAM algorithms is the incorrect graph structure used for map optimization. Two main types of errors are false positive and false negative associations produced by the SLAM front-end. False positive associations arise when environment scans are registered as belonging to the same place while they are unrelated in the real world, whereas false negative associations fail to match environment scans related in the real world.

This section provides a more detailed review of challenging environments and existing solutions. First, Section 2.3.1 gives some real-world examples of challenging environments. Next, Section 2.3.2 reviews the existing solutions that address algorithmic SLAM failures.

2.3.1 Challenging environments

One of the main challenges of robotics is to develop robots that can navigate the world alongside humans. We rarely acknowledge the fact that man-made environments are challenging by their nature because they contain a lot of repetitive elements and similar-looking places. We have built sophisticated infrastructure to help us navigate the world around us. This infrastructure includes global positioning systems, division of territories into countries, cities and streets with unique identifiers (names and postal codes), labeling systems in indoor environments (supermarkets, warehouses, libraries), road signs and signals as well as many other elements. However, often times this infrastructure is beyond reach for the modern day robots. Although researchers push in the direction of more high-level, semantic understanding of the environment [38], most of the current state-of-the-art robots rely on geometric maps for navigation. What are some examples of challenging environments that can lead to the failure of SLAM systems?

First of all, environments with repetitive and indistinguishable elements make robots susceptible to perceptual aliasing and false positive associations [39]. Some examples are urban outdoor environments with similar building blocks and large

indoor environments such as warehouses and supermarkets with repetitive shelves of products. Even though the problem of false positive loop closures could in principle be mitigated by using global positioning sensors such as RTK GNSS receivers but in practice, GNSS signal is often unreliable when occluded by tall buildings and trees. Additionally, global positioning systems are not available for indoor environments at all.

Another navigational challenge for robots is the environments with few distinguishable features that produce false negative associations which means no loop closures. Examples of this kind of environments are open flat outdoor fields with no natural or artificial landmarks above the ground and long indoor corridors of featureless walls. In these environments proprioceptive sensors remain the main source of data but the lack of external landmarks and the associated loop closures leads to the accumulation of the dead-reckoning drift error [40].

Finally, changes in environments both short-term (moving people and objects, road works, bad weather, seasonal changes) and long-term (construction and demolition of buildings, natural disasters) break the assumption of the static world often times used in modern SLAM systems.

2.3.2 Existing solutions

The nature of algorithmic SLAM failures dependent on the incorrect graph structure makes it logical to divide existing solutions based on the different parts of the SLAM pipeline. The existing solutions are presented below according to where the interventions are applied.

Front-end

The front-end solutions focus on the correct detection and validation of map features using different techniques for different sensors [41]–[43]. The first obvious solution for feature-based matching methods is to compare the current measurement with all previously detected features but it quickly becomes infeasible with growing number of measurements. Bag-of-words models [44] and hierarchical vocabulary trees [45] address the issue of the growing search space by combining features into larger groups, which makes the lookup more efficient. However, these methods fail to match features with large illumination changes. Another group of methods address this issue by associating sequences instead of individual features [46], unifying different features into one representation [47] or incorporating spatial information [48].

While the above mentioned methods address the correct loop closure detection, RANSAC [49] and residual error check methods validate loop closures after they have been matched. Environment dynamics is addressed by keeping track of changes through several maps of the same location [50] or keeping a time-variant parameter on one map [51]. However, perceptual aliasing in challenging environments makes it inevitable that the front-end feeds false positive loop closures into the back-end. This leads us to the back-end solutions.

Back-end

The back-end solutions are better positioned to deal with wrong data associations because it receives the whole graph data and several solutions leverage this advantage.

One group of methods address the issue by checking the residual error during optimization [52]–[54] and effectively rejecting false positive or selecting only true positive loop closures. Another method is to cross-check loop closures with odometry measurements before optimization [55]. Yet another method is to add new constraints and variables (i.e. graph topology) into the estimation [5]. Nonetheless, even the back-end solutions are fragile in challenging environments [4].

Map representation

The last place in the SLAM pipeline to address algorithmic SLAM failures is the graph interface itself. When automated solutions in the front-end and the back-end fail, the third alternative is to have a human interfere between them to manually edit the graph underlying the map. There were several recent proposals for human-in-the-loop solutions, which are presented in Table 2 and discussed below.

Table 2: The existing human-in-the-loop solutions

ID	Name	Year	Paper
SL-1	MAGIC 2010: Team Michigan	2013	[56]
SL-2	MAGIC 2010: Team MAGICian	2012	[57], [58]
SL-3	A-SLAM	2018	[59]
SL-4	Human-in-the-Loop SLAM	2018	[60]
SL-5	Interactive SLAM	2020	[61]
SL-6	Interactive 3D Graph SLAM	2020	[62]

The earliest human-in-the-loop solutions SL-1 and SL-2 from Table 2 were developed and used in Multi Autonomous Ground-robotic International Challenge (MAGIC) held in Adelaide, Australia in November 2010 by two different teams. One of the tasks participating teams had to achieve was to map a large (500m×500m) urban area, consisting of both indoor and outdoor segments, completely and accurately within the time limit of 3 h 30 min using a team of robots. This task required them to develop and use a human machine interface application to allow human operators overseeing the robot teams to intervene when needed. One of the interventions identified by the teams was to recognize and correct mapping errors during the robot operation online. Team Michigan, a collaboration between the University of Michigan's APRIL Lab and Soar Technology, Inc., developed a user interface that allowed human operators to monitor the global map, roll back false positive automatic scan matches, add additional manual scan matches and resume the automatic mapping process. This approach required limited human interventions and allowed Team Michigan eventually win the competition. Team MAGICian representing universities and companies from Australia developed a similar human machine interface (HMI) solution, which allowed human operators to monitor the global map, adjust robot poses in the map, translate and rotate created graph nodes and add corresponding relative constraints. Both solutions used 2D representation of the environment and graph based SLAM implementation. Given the constraints of the challenge, the focus of these two solutions was on providing a user interface to fix map errors online by highly trained human operators. For more details about

the challenge, competing teams and their approaches refer to the special issue of the Journal of Field Robotics [63]–[66] dedicated to the competition.

The next solution SL-3 was developed for a human operator operating online as well but only for a single robot. Solution SL-3 is the only solution from Table 2 that uses particle filter based SLAM approach [67]. Other distinguishing features of solution SL-3 are the use of the augmented reality headset to visualize the output of the robot mapping process and correcting both robot poses and environment map itself. Solution SL-4 introduces an offline post-processing approach different from the online approach used in the previous solutions. The offline approach allows to decouple data gathering and map creation stages. Other features of solution SL-4 are the introduction of a new human correction constraint and human input interpretation [60]. Solution SL-5 introduced a more intuitive graphical user interface to correct for submap poses [61].

Finally, the most recent solution SL-6 used 3D representation of the environment for the first time. Two dimensional representation of the environment is not able to capture some aspects of the 3D environment structure inside tunnels and under bridges as well as it is not sufficient for reliable collision avoidance and path planning in 3D world [68], [69]. Another contribution of solution SL-6 is the introduction of the pose edge refinement and plane-based map correction techniques [62]. Comparison of some of the parameters is presented in Table 3.

Table 3: Comparison of the existing human-in-the-loop solutions

ID	Type	Map	Robots	SLAM	SLAM back-end
SL-1	Online	2D	Multiple	Graph	Custom
SL-2	Online	2D	Multiple	Graph	SPA [29]
SL-3	Online	2D	Single	Particle filter	Gmapping [67]
SL-4	Offline	2D	Single	Graph	Ceres Solver [70]
SL-5	Offline	2D	Single	Graph	Google Cartographer [71]
SL-6	Offline	3D	Single	Graph	g^2o [28]

Chapter 2 reviewed the existing solutions to algorithmic SLAM failures across all parts of the SLAM pipeline and identified that the most robust solutions at the moment still require human involvement. The following Chapter 3 identifies system requirements for the design of such a solution in Chapter 4.

3 Requirements analysis

In order to answer one of the research questions (RQ1) of this thesis, requirements analysis work was conducted to list, classify and prioritize requirements. A structured approach for this work ensures that all relevant stakeholder concerns and needs are addressed in the software application. Two main sources of information used here were unstructured interviews with several stakeholders and feature analysis of existing solutions described in Section 2.2.3. Although stakeholder interviews are enough to guide the design and development of the proposed solution, more general considerations were also extracted from prior work to ensure a wider perspective on the problem. Therefore, the requirements analysis phase of this thesis pursues two related goals:

- developing a list of stakeholders and their needs;
- framing the system design and development based on the relevant system requirements.

As a consequence of the first goal, this thesis defines two sets of stakeholders: a baseline set presented in Section 3.1 and an extended set presented in Appendix A. The former is relevant for this thesis and the following system requirements while the latter could be used to guide future development and changes.

The methodology for requirements analysis is based on ISO/IEC/IEEE 29148:2018 (E) standard, *Systems and software engineering — Life cycle processes — Requirements engineering* [72] referred as ISO 29148 and NASA System Engineering Handbook [73] referred as NASA SE Handbook. ISO 29148 was used as the main framework and NASA SE Handbook as the supporting reference material.

ISO 29148 distinguishes three levels of requirements hierarchy as well three requirements engineering processes related to them:

- *Business or mission analysis* process that defines *business requirements* on the business management level;
- *Stakeholder needs and requirements definition* process that defines *stakeholder requirements* on the business operational level. NASA SE Handbook defines an equivalent process as *stakeholder expectations definition*;
- *System requirements definition* process that defines *system requirements* on the system/software level. NASA SE Handbook defines an equivalent process as *technical requirements definition*.

The business and mission analysis process defines problem and solution spaces and the preferred solution class out of all candidate alternatives. These tasks were accomplished in Chapter 2. The problem space is described and constrained to algorithmic SLAM failures in Section 2.3.1, the solution space and the preferred solution class of human enhanced SLAM graph solutions are defined in Section 2.3.2.

Chapter 3 continues with the remaining two requirements engineering processes as follows. Section 3.1 describes stakeholder needs and requirements definition process, which consists of stakeholder identification, definition of their needs and acceptance

criteria, prioritization and verification of stakeholder requirements. Section 3.2 follows the system requirements definition process, which includes definition of system requirements and design constraints, their analysis and definition of performance measures. System requirements defined here are later used in Chapter 4 for the system design.

3.1 Stakeholders and their needs

This thesis defines a concept of generalized robotics company (from here on - the Company) and uses the following assumptions for developing the stakeholder requirements:

- the Company is a company developing mobile robots (including autonomous vehicles) or full-stack software solutions for them, using map-based navigation and graph-based SLAM algorithms for building environment maps offline;
- the Company faces a challenge of poor map estimates in challenging environments;
- the Company would like to have a robust solution for this problem;
- this thesis presents a general solution for the Company.

Stakeholders

Following ISO 29148 definition of a minimum set of stakeholder groups for the system or software development project[2], this thesis identified two sets of stakeholders: baseline and extended. The baseline stakeholder set includes User, Acquirer and Developer groups whose interests and concerns are essential for the development of the proposed software application. The extended set includes Operator, Maintainer, Regulatory Authority groups whose interests and concerns are less pronounced at the beginning but would become more important at later stages when the proposed software application is released as a product. The stakeholder groups, software systems and their relationships is illustrated on Figure 6.

Table 4: The baseline set of stakeholders

ID	Group	Roles
S-1	User	Robotics Engineer, Autonomous Vehicle Engineer
S-2	Acquirer	Chief Technology Officer
S-3	Developer	Software Engineer

This thesis uses the baseline set of stakeholders (Table 4) and their needs to develop system requirements in Section 3.2. The User group (S-1) in the context of this thesis includes Robotics Engineers within the Company that would use the proposed solution to fix poor SLAM maps. The Acquirer group (S-2) consists of the Chief Technology Officer of the Company that would like to acquire a solution

[2]Section 5.2.2 of ISO 29148.

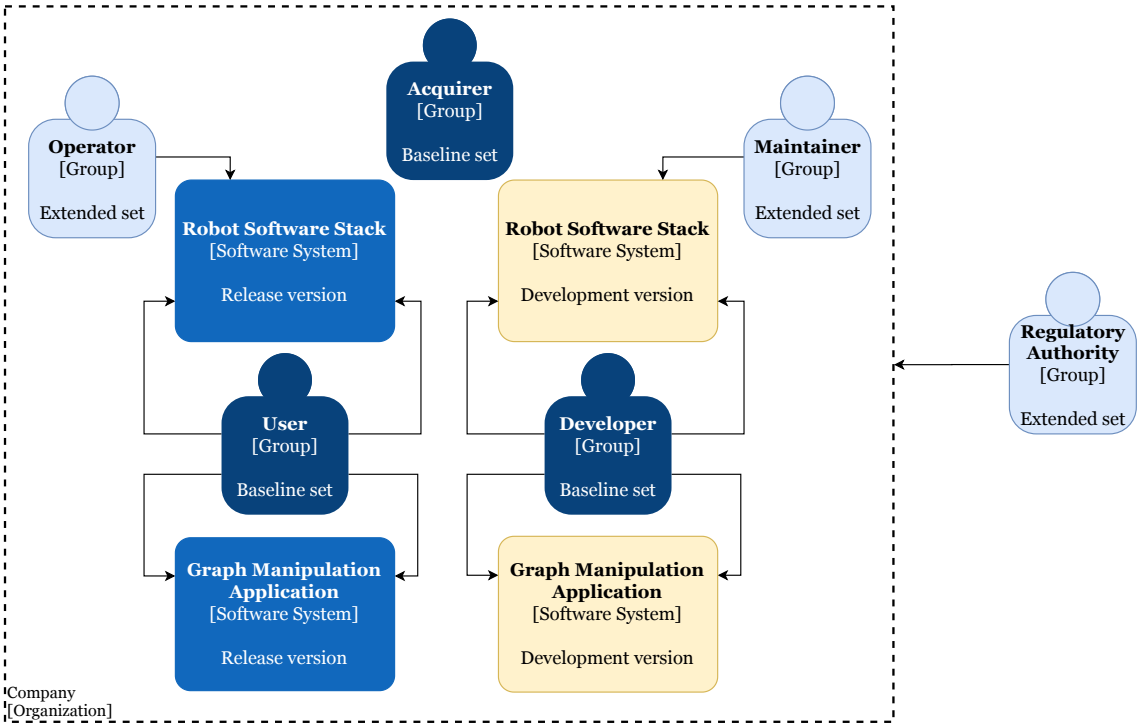

Figure 6: Stakeholders identified for the proposed software application. They are divided into baseline and extended sets reflecting their importance at different software development stages.

for fixing poor SLAM maps. The Developer group (S-3) is concerned with the development of the proposed solution for the Company.

The extended set of stakeholders adds three more stakeholder groups. The Operator group (S-4) acts as a secondary user and is interested in operating a robot safely using the output maps from the proposed solution. The main concern of the Maintainer group (S-5) is to have low maintenance software that is easy to extend. The Regulatory Authority group (S-6) is perhaps the most important extension because their interests and concerns are coming from outside of the Company and include conformance to state and local legislation, industry standards, and might introduce new design constraints. Table A1 summarizes all stakeholder groups and their high-level expectations. Detailed identification of needs of the extended set of stakeholders is out of the scope of this work.

Stakeholder needs

The next step after identifying all stakeholders is to define their needs. This is done through several iterations of discussions with stakeholders to identify needs, prioritize the most important of them, and verify their correctness and validity. The result is summarized in Table 5 in the form of user stories[74]. Note that a new attribute called Type is introduced to distinguish between needs, user stories and system constraints. They are roughly divided based on three groups of stakeholders. User stories describe what stakeholders expect the system to do while needs and system constraints describe how it is done.

Table 5: Stakeholder needs.

ID	Type	Description
N-1	Need	As a acquirer, I want to have a tool that is effective at discovering and fixing the map errors so that a robot can navigate safely and reliably using the corrected map
N-2	Need	As a acquirer, I want to have maps corrected separately from the main task of navigation so that I minimize the risk of online navigation failures
N-3	Need	As a acquirer, I want to have a tool use the same map format as the rest of the software stack so that developers do not have to change the existing software stack
N-4	Constraint	As a acquirer, I want to have a modular software architecture so that developers can modify and replace modules in isolation
N-5	Constraint	As a acquirer, I want to have users use a tool on any device so that they do not have to install additional software
N-6	Constraint	As a developer, I want to reuse existing communication libraries so that I can develop faster
N-7	Constraint	As a developer, I want to reuse the same map storage type as the existing software stack so that I can develop faster
N-8	Constraint	As a developer, I want to reuse existing graph optimization libraries so that I can develop faster
N-9	Constraint	As a developer, I want to reuse existing submap registration algorithms/libraries so that I can develop faster
N-10	User story	As a user, I want to load the map files created on another machine so that I can use any available machine
N-11	User story	As a user, I want to save the corrected map files in the compatible format so that I can use the corrected map for robot localization
N-12	User story	As a user, I want to visualize the map's underlying graph and point clouds so that I can inspect the mapping result visually
N-13	User story	As a user, I want to visualize map's components in layers so that I can switch them on or off
N-14	User story	As a user, I want to clearly see which map objects are selected and have their attributes clearly displayed as well so that manipulations that I do are predictable
N-15	User story	As a user, I want to clearly see which map objects were changed so that I can easily compare the results
N-16	User story	As a user, I want to have different types of point cloud parameters (intensity, RGB) correctly visualized so that I can leverage all of the available information to visualize point clouds
N-17	User story	As a user, I want to clearly see map errors highlighted so that I can quickly start fixing them
N-18	User story	As a user, I want to translate and rotate graph nodes so that I can adjust and align their poses
N-19	User story	As a user, I want to change the constraint values and add new relative constraints so that I can constrain, relax/tighten or remove constraints between nodes and on nodes

ID	Type	Description
N-20	User story	As a user, I want to see the origin and the end of the whole map so that I can quickly identify the beginning and the end of mapping trajectory
N-21	User story	As a user, I want to have moved submaps automatically registered so that they are aligned automatically
N-22	User story	As a user, I want to register two submaps with newly added relative constraint so that they are aligned
N-23	User story	As a user, I want to execute graph optimization so that the global map is optimized with new graph parameters
N-24	User story	As a user, I want to be notified when back-end services fail so that I know the reason for failure

Acceptance criteria

The last step in the stakeholder needs and requirements definition process is to define acceptance criteria for each need to have a way of measuring if the need has been satisfied in the proposed solution. The acceptance criteria are formulated as use cases that have to be covered using the proposed solution. The result is summarized in Table 6.

Table 6: Acceptance criteria for stakeholder needs.

ID	Acceptance criteria
N-1	User has a way of evaluating and fixing a map quality
N-2	User corrects maps offline
N-3	User can reuse existing maps
N-4	Developer can modfiy and replace modules without affecting other modules
N-5	User can use the application from any operating system without installing additional software
N-6	Developer can reuse off-the-shelf libraries
N-7	Developer can reuse existing map storage type
N-8	Developer can reuse existing code for graph optimization
N-9	Developer can reuse existing code for submap registration
N-10	User can open maps created on another machine
N-11	User can save modified maps and use them on another machine
N-12	User can see both graph and point clouds
N-13	User can choose which elements to show and hide
N-14	User can select objects
N-15	User can see changes
N-16	User can load point clouds with different parameters
N-17	User can see map errors
N-18	User can change position and orientation of graph nodes
N-19	User can add and remove graph edges User can change graph edge values
N-20	User can see mapping trajectory start and end

ID	Acceptance criteria
N-21	Moved nodes are registered automatically
N-22	User can manually register two submaps
N-23	User can execute graph optimization
N-24	User is notified with meaningful messages when back-end services fail

3.2 System requirements

This section builds on the identified stakeholder needs from Section 3.1 to define system requirements, design constraints and their performance measures that will guide design choices in Chapter 4.

Requirements

System requirements are formulated using the requirement language criteria specified in Section 5.2.7 of ISO 29148. Additionally, this thesis analyzed existing solutions from Section 2.3.2 to identify system requirements similar to the ones defined in this work. This helped to prioritize and select the most generally useful system requirements. The result is summarized in Table 7 with corresponding references to existing solutions.

Table 7: Requirements

ID	Type	Requirement	Existing solution
R-1	Non-Functional	The system shall be effective at discovering map errors	
R-2	Non-Functional	The system shall be effective at correcting map errors	SL-1,SL-2, SL-3,SL-4, SL-5,SL-6
R-3	Non-Functional	The system shall be used in offline SLAM process	SL-4,SL-5, SL-6
R-4	Non-Functional	The system shall use the existing map format	
R-5	Non-Functional	The system software architecture shall be modular	
R-6	Non-Functional	The system shall be available for use on Linux, Windows, Mac platforms	
R-7	Non-Functional	The system shall use existing communication libraries	
R-8	Non-Functional	The system shall save map files in the same storage type as used by existing SLAM process	
R-9	Non-Functional	The system shall use existing graph optimization libraries	
R-10	Non-Functional	The system shall use existing submap registration algorithms/libraries	

ID	Type	Requirement	Existing solution
R-11	Functional	The system shall allow users to load existing map files	SL-6
R-12	Functional	The system shall allow users to close the current map project	SL-6
R-13	Functional	The system shall allow users to save the opened map into a file in the compatible format	SL-6
R-14	Functional	The system shall visualize graph components	SL-2,SL-4, SL-6
R-15	Functional	The system shall visualize point clouds	SL-4,SL-6
R-16	Functional	The system shall load graph components and point clouds into separate layers	SL-2
R-17	Functional	The system shall indicate the selected object and its attributes visually	SL-2
R-18	Functional	The system shall indicate the changed objects visually	SL-2,SL-6
R-19	Functional	The system shall allow users to change point cloud's point size	
R-20	Functional	The system shall allow users to change point cloud's point color	
R-21	Functional	The system shall indicate mapping errors visually	
R-22	Functional	The system shall allow users to move existing graph nodes	SL-2
R-23	Functional	The system shall allow users to remove existing graph nodes	SL-2
R-24	Functional	The system shall allow users to add new edges (relative constraints) between graph nodes	SL-1,SL-5, SL-6
R-25	Functional	The system shall allow users to change existing edges	
R-26	Functional	The system shall allow users to remove existing edges	SL-6
R-27	Functional	The system shall indicate map origin visually	
R-28	Functional	The system shall indicate map trajectory end visually	
R-29	Functional	The system shall register two submaps automatically if they have existing map registration constraint and one of them is moved	
R-30	Functional	The system shall allow users to register two submaps manually if a user creates a new edge (relative constraint)	SL-1,SL-5, SL-6
R-31	Functional	The system shall allow users to run graph optimization	SL-6
R-32	Functional	The system shall notify a user of back-end service failures	SL-6

Performance measures

Performance measures for system requirements are formulated as tests for each requirement in Table 8 while the results of these tests are presented in Chapter 5.

Table 8: Tests against the requirements

ID	Test	Expected result
R-1	Discover map errors visually	User can discover map errors visually
R-2	Fix indoor map	User can fix indoor map
R-2	Fix outdoor map	User can fix outdoor map
R-3	Confirm that the system is used in offline SLAM mode	The system works offline from the main SLAM process
R-4	Confirm that the system uses the existing map format	The system uses the existing map format
R-5	Confirm that the system architecture is modular	The system architecture is modular
R-6	Confirm that the system is platform-independent	The system is platform-independent
R-7	Confirm that the system uses existing communication libraries	The system uses the existing communication library
R-8	Confirm that the system uses the existing storage type	The system uses the existing storage type
R-9	Confirm that the system uses existing graph optimization library	The system uses the existing graph optimization library
R-10	Confirm that the system uses existing submap registration algorithm/library	The system uses the existing submap registration algorithm
R-11	Load existing map files	User can load existing map files
R-12	Close currently opened map	User can close currently opened map
R-13	Save currently opened map	User can save currently opened map
R-14	Confirm graph components visualized correctly	The system displays graph components correctly
R-15	Confirm point clouds visualized correctly	The system displays point clouds correctly
R-16	Switch on/off layers for graph components and point clouds	User can switch on/off map layers
R-17	Select an object and verify that its attributes are displayed	The system displays attributes of the selected object
R-18	Move and rotate graph nodes to see them changed	The system indicates changed graph nodes correctly
R-18	Run map optimization to see objects change their poses	The system indicates changed graph nodes correctly
R-19	Change point cloud's point size	User can change point cloud's point size

ID	Test	Expected result
R-20	Change point cloud's point color	User can change point cloud's point color
R-21	Confirm mapping errors visualized correctly	The system indicates mapping errors
R-22	Move a graph node	User can move and rotate graph nodes
R-23	Remove a graph node	User can remove graph nodes
R-24	Add a new relative constraint between two nodes	User can add a new relative constraint between two nodes
R-25	Change covariance values on an existing relative constraint	User can change covariance values on an existing relative constraint
R-26	Remove an existing relative constraint	User can remove an existing relative constraint
R-27	Confirm a map origin visualized correctly	The system indicates a map origin
R-28	Confirm a map trajectory end visualized correctly	The system indicates a map trajectory end
R-29	Move a graph node with a map matching constraint	The system registers moved graph nodes with a map matching constraint automatically
R-30	Create a new relative constraint and register it manually	User can register a new relative constraint manually
R-31	Run graph optimization	User can run map optimization
R-32	Confirm that the system notifies of back-end service failures	The system notifies of back-end service failures

Chapter 3 identified stakeholder and system requirements for a human-in-the-loop SLAM solution and the following Chapter 4 introduces the design of Graph Manipulation Application, which is a software implementation of the aforementioned requirements.

4 Graph manipulation application

The system was designed based on stakeholder and system requirements presented in Chapter 3. The main approach to the system design was the layered software architecture using C4 notation[75]. This approach allows to gradually increase the complexity of the software architecture with the goal of refining and improving the initial design with each iteration. Another benefit is that the different levels of detail allow communicating software architecture efficiently and effectively to different groups of stakeholders. Even though software architecture diagrams on each level are useful for several stakeholders, each of them has a different focus and the main intended audience. More details are provided in Sections 4.1- 4.3 describing levels of the software architecture.

Chapter 4 details design decisions taken and their rationale. Section 4.1 presents system level architecture. Section 4.2 zoom in on subsystem level. Section 4.3 provides more details about the system components and relations between them.

4.1 System level

The main goal of the system level layer of the software architecture is to define the context surrounding the system in the form of other systems and actors. The main intended audience for the diagram on Figure 7 is Acquirer stakeholder group described in Section 3.1. The system level software architecture describes how the system is integrated into the overall software stack and what are the main interfaces between them.

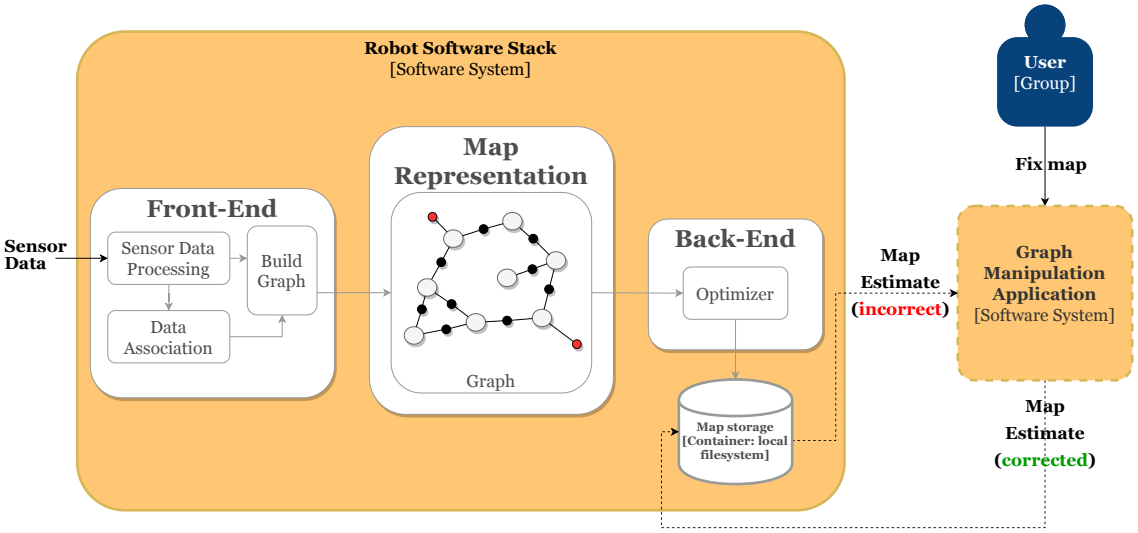

Figure 7: System context diagram (C4 model) for Graph Manipulation Application.

Figure 7 shows the three main parts of the system level software architecture: Robot Software Stack, Graph Manipulation Application and User. Robot Software Stack represents the existing robot software, which contains the SLAM pipeline presented in Section 2.2.2. During the mapping stage, the output map estimate is

saved in the local file system for later use in the localization stage. However, the output map is indicated as incorrect and needs further validation and post-processing to ensure safe and robust navigation of the robot. This is where the proposed Graph Manipulation Application comes in to help validate and fix the initial map estimate. Human experts using the proposed system are indicated as a User group.

The diagram on Figure 7 addresses non-functional system requirements R-1, R-2, R-3 and R-4 from Table 7 and corresponding stakeholder requirements of the type Need from Table 5: N-1 (an effective tool to discover and fix map errors), N-2 (offline map correction process), N-3 (reusing the map format).

The proposed system design is detached from online robot operations and assumes offline mapping process to allow for offline validation and post-processing of SLAM output map. As a consequence, the interfaces of Graph Manipulation Application for the input and output map estimates are not dependent on Robot Software Stack system implementation and depend only on the saved map format, which is defined separately. This makes the Graph Manipulation Application stable and universal across different robots.

4.2 Subsystem level

The subsystem level software architecture helps to define high-level internal parts (called containers according to C4 notation) of the system and their functional responsibilities. The main intended audience for the diagram on Figure 8 is the User stakeholder group described in Section 3.1. The subsystem level software architecture describes how the internal parts communicate with each other and the major technology choices.

There are four main modules of the subsystem level software architecture:

1. Web application module provides the main user interface to manipulate the graph;
2. Communication module provides transport capabilities to connect the front-end to the back-end;
3. Services module contains back-end services that process graph data. Here the application performs two computationally intensive operations of map registration and map optimization (see Section 4.3 for details);
4. Map storage module saves map data for later use.

There are several early design choices made on this level of abstraction, which address the non-functional system requirements R-5 through R-8 defined in Section 3.2.

Web Application. It was decided early on to develop a web-based solution to satisfy the non-functional system requirement R-6. This would allow us to use it from any machine without the need to install additional software. Also the choice of web application architecture allows separating user interface in the front-end[3] and

[3]Note that terms front-end and back-end are used in different contexts in Chapters 2 and 4. The former refers to SLAM pipeline and the latter to web-based software architecture.

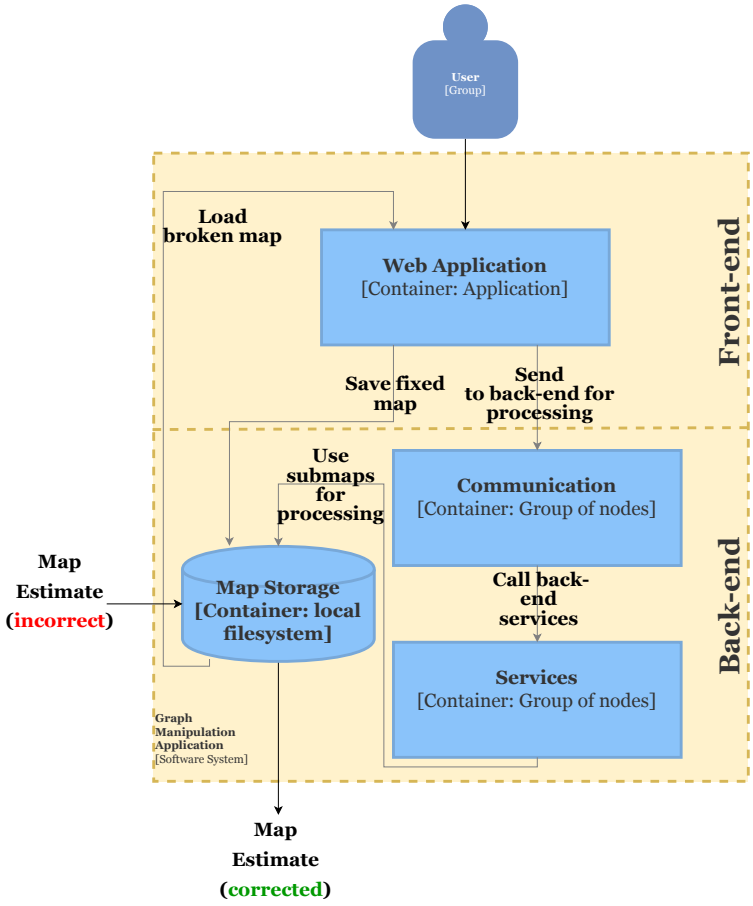

Figure 8: Container diagram (C4 model) for Graph Manipulation Application.

graph processing services in the back-end, which satisfies the non-functional system requirement R-5.

Services. The Robot Operating System (ROS)[76] middleware is used as an interface for the communication with back-end services. ROS is a de facto standard framework for robotics development and satisfies the non-functional system requirement R-7.

Communication. The standard way to communicate between ROS back-end and web front-end is to use WebSocket communications protocol. WebSocket provides a two-way communication channel that does not require opening a connection every time a client needs a back-end service. This satisfies the non-functional system requirement R-7.

Map storage. A local file system is used for map storage because the existing implementation of Robot Software Stack system saves maps this way and this satisfies the non-functional system requirement R-8.

4.3 System component level

The next level of the software architecture is the system component level with the goal of identifying the major building blocks of subsystems and their implementation details. The main intended audience for the diagram on Figure 9 is the Developer stakeholder group described in Section 3.1. The system component level software architecture describes how the internal parts are integrated with each other to realize the functionality of subsystems and how they are implemented.

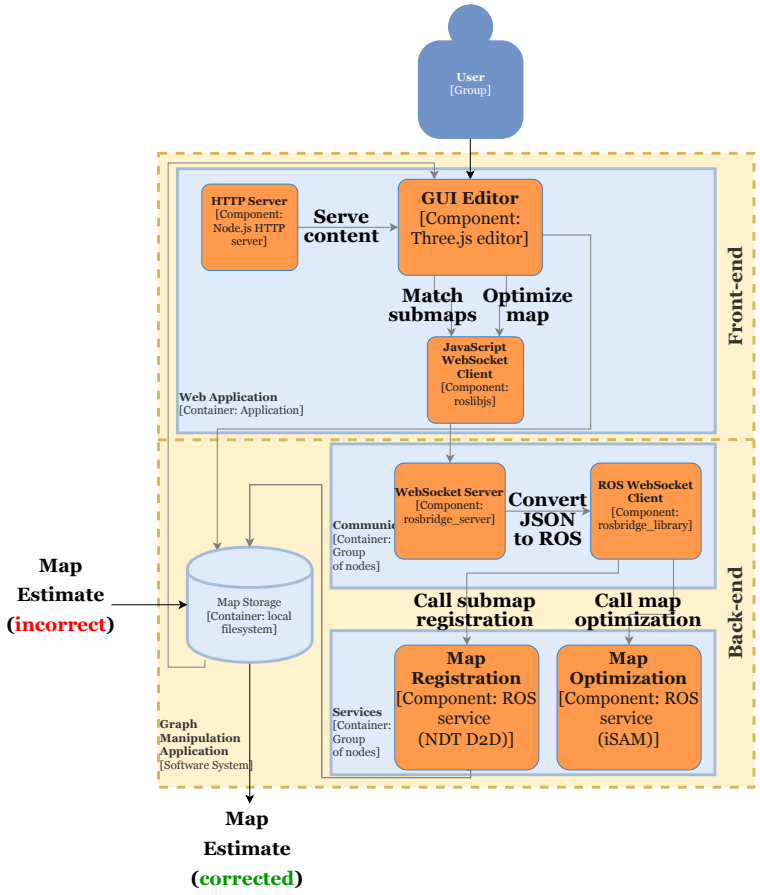

Figure 9: Component diagram (C4 model) for Graph Manipulation Application.

The system component level consists of seven system components that address system requirements from Section 3.2 and contain implementation details. Four components out of seven use off-the-shelf packages.

4.3.1 Front-end

The front-end consists of three system components: GUI Editor, HTTP Server and JavaScript WebSocket Client. The built-in Node.js HTTP module is used as HTTP Server to serve content to the browser and roslibjs JavaScript library (part of `rosbridge_suite`[77] metapackage) as JavaScript WebSocket Client to

communicate with ROS back-end from the browser. Both components are used as-is without modification.

GUI Editor

GUI Editor is implemented using the Three.js [78] library and editor. Three.js is a JavaScript library for visualizing 3D content in the browser using WebGL renderer. The main showcase application for Three.js features is the Three.js editor, which provides a good baseline for Graph Manipulation Application browser front-end to work with 3D map graph.

Table 9 maps system requirements defined in Section 3.2 to out-of-the-box Three.js editor features and extensions developed for this thesis.

As we can see from Table 9, Three.js editor provides a good baseline implementation for addressing the system requirements. Figure 10 illustrates GUI Editor of the Graph Manipulation Application adapted from the original Three.js editor.

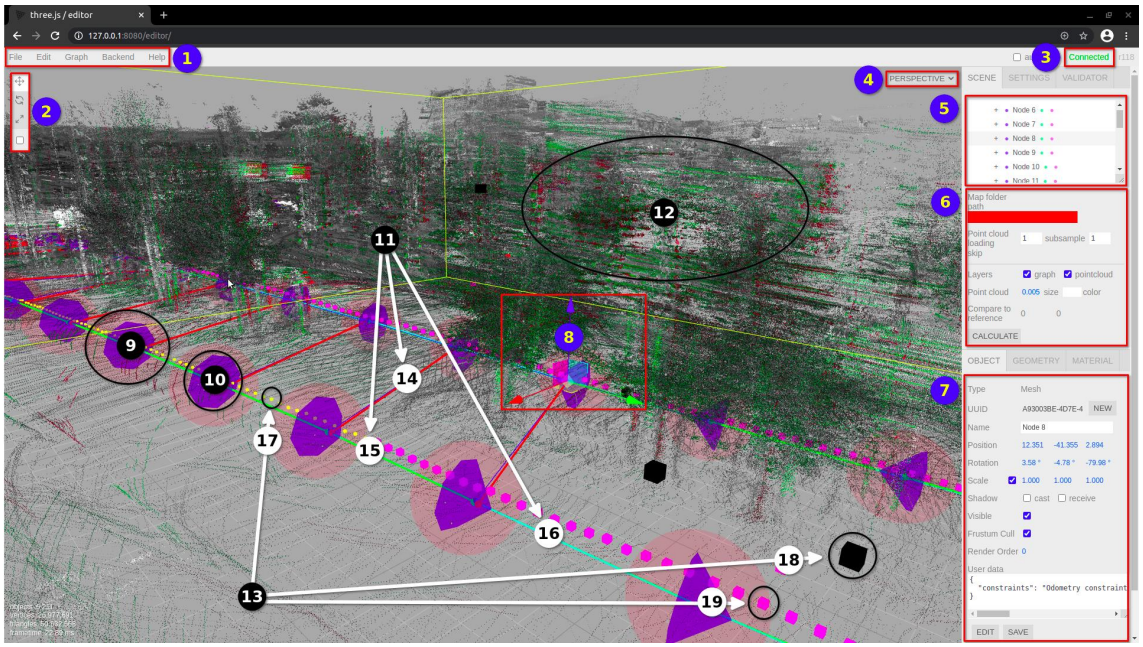

Figure 10: GUI editor of Graph Manipulation Application. 1 - top menu, 2 - object controls mode, 3 - back-end connection status, 4 - camera views, 5 - object tree, 6 - scene parameters, 7 - object parameters, 8 - object transform controls, 9 - absolute constraint, 10 - graph node, 11 - relative constraints (14 - loop closure, 15 - odometry constraint, 16 - map matching constraint), 12 - point cloud, 13 - GNSS-RTK quality (17 - full RTK (good), 18 - other (bad), 19 - float RTK (weak)).

4.3.2 Back-end

The back-end consists of four system components: WebSocket Server, ROS WebSocket Client, Map Registration and Map Optimization. The rosbridge protocol and implementation are used for communication. `rosbridge_server` [79] is used as WebSocket Server to provide WebSocket connection between the browser and the back-end while `rosbridge_library` [80] is used as ROS WebSocket Client to

Table 9: Mapping functional requirements to Three.js features

ID	Three.js feature	Implemented feature
R-11	File loaders for .txt and .pcd files	File loader for iSAM graph file, small modifications to PCDLoader
R-12	Menu option 'New', which clears the editor	Adapted for 'Close' menu option
R-13	File exporters for different file types	File exporter for submap origins
R-14	Built-in	
R-15	Built-in	
R-16	Built-in	
R-17	Built-in	
R-18	Built-in visual changes	
R-19	Not implemented directly but provides indirect built-in mechanisms	Implemented through the side panel menu
R-20	Not implemented directly but provides indirect built-in mechanisms	Implemented through the side panel menu
R-21	Built-in visual map error discovery	
R-22	Built-in TransformControls to move objects	Adapted
R-23	Built-in object tree removal	
R-24	Not implemented directly but provides indirect built-in mechanisms	Implemented through graph edge context menu
R-25	Not implemented directly but provides indirect built-in mechanisms	Implemented through User data parameter on the graph edge.
R-26	Built-in object tree removal	Added removal through graph edge context menu
R-27	Not implemented directly but provides indirect built-in mechanisms	Implemented using color. Trajectory start graph node is colored red.
R-28	Not implemented directly but provides indirect built-in mechanisms	Implemented using color. Trajectory end graph node is colored green.
R-29	Not implemented directly but provides indirect built-in mechanisms	Implemented. Submap constraints saved on graph nodes
R-30	Not implemented directly but provides indirect built-in mechanisms	Implemented through graph edge context menu
R-31	Not implemented	Implemented through the top menu option 'Map optimization'
R-32	Not implemented	Not implemented in the GUI Editor itself, but available in the browser Web Console and back-end console

convert messages from JSON to ROS format and back. Both components are used as-is without modification.

Map Registration

Map Registration is implemented as a ROS service wrapper node using C++. This system component matches submaps in NDT-OM format using the D2D-NDT

algorithm[19]. The core algorithm is a part of the existing software stack and this system component provides it as a service for the front-end. Map Registration addresses the following functional system requirements from Table 7:

- R-10 - using the existing submap registration algorithm - D2D-NDT algorithm;
- R-29 - automatic submap registration - called automatically when graph nodes are moved;
- R-30 - manual submap registration - user can register two submaps using the graph edge context menu.

Map Optimization

Map Optimization is implemented as a ROS service node using C++ and iSAM library. This system component constructs a factor graph in the back-end based on the data from the front-end and optimizes the global map using iSAM built-in method[23]. Map Optimization addresses the following functional system requirements from Table 7:

- R-9 - using the existing graph optimization library - iSAM library;
- R-31 - run graph optimization - user can run map optimization from the top menu.

Chapter 4 defined the software architecture of Graph Manipulation Application based on the system requirements from Chapter 3. The following Chapter 5 presents the evaluation methodology and results of experiments for validating the effectiveness of the proposed solution.

5 Experiments and results

Chapter 5 describes experiments performed with the developed software application to answer the question of effectiveness of the proposed solution (RQ2) as well as the question of the most effective user manipulations (RQ3). Section 5.1 defines a methodology used to evaluate and compare results. Sections 5.3 and 5.2 describe experiments and results in challenging outdoor and indoor environments. Finally, Section 5.4 demonstrates results for tests against system requirements defined in Chapter 3.

5.1 Evaluation methodology

There are several options when assessing the effectiveness of the proposed solution. First, we can think of evaluating a map estimate produced with Graph Manipulation Application by a user. As we mentioned earlier humans are good at processing the visual information and visual inspection is the first intuitive way of evaluating the quality of the produced map estimate. However, there is a problem of changes so small that they are not detectable by the naked eye. This leads to a second consideration which requires a more accurate and objective evaluation method. Here we developed two additional approaches used in this thesis. Below is the description of all three evaluation methods and their advantages and drawbacks.

5.1.1 Visual inspection by a human user

The initial idea for evaluation of results was to conduct a visual inspection of produced maps. This method requires a human user to visually inspect the produced maps for inconsistencies of how point clouds match and reflect the real-world environment. A human user can detect map segments, where point clouds do not align and move the corresponding graph nodes to align them better. This would produce better local constraints. Then the user calls the back-end map optimization service to process a new input graph to produce a better global estimate. We define three types of results based on visual inspection:

- Distorted - a map estimate is visibly distorted (alignment error is large);
- Partially corrected - a map estimate is close to the correct estimate but contains smaller distortions (drift, double walls, straight corridors are bent);
- Corrected - a map estimate is visibly correct.

The result of type Corrected is the aim of graph manipulations. The advantages of this evaluation method include high interpretability of results and low effort since it does not have additional requirements. However, during the experimentation phase, it became evident that this is not enough to distinguish between two visibly correct maps in cases, where the changes are so small that they are not detectable by the naked eye. Therefore, two new approaches were explored and implemented.

5.1.2 Distance to a reference map graph

The first alternative evaluation method to visual inspection is calculating the distance to the reference map graph nodes. We have two maps: the map after correcting user manipulations and the reference map, which we know is correct. We calculate a distance from corrected map graph nodes to the corresponding nodes in the reference map graph and sum up distances for each node in the map we are evaluating to get the overall distance score between two maps. Since node poses consist of position and orientation components, the distance score is broken down into two components: translation and rotation distance (Figure 11). This allows for more fine-grained evaluation and comparison.

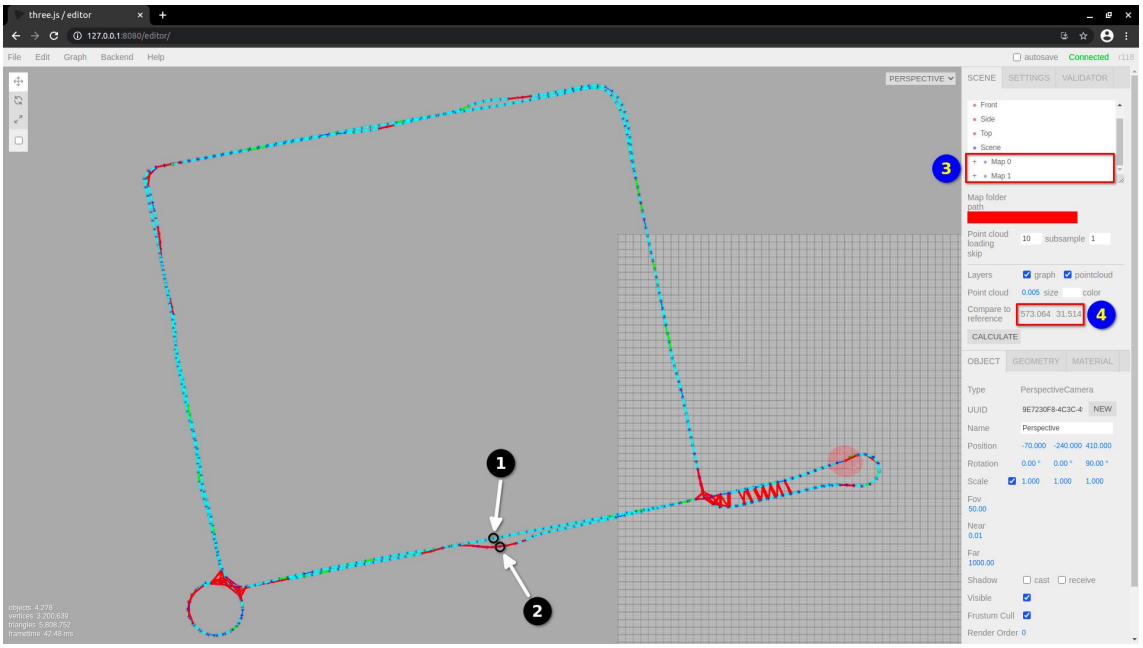

Figure 11: Total distance score components are obtained by summing up distance components for each pair of corresponding nodes. 1 - graph node from Map 1 (reference), 2 - corresponding graph node from Map 0 (evaluated), 3 - both maps in the object tree, 4 - calculated distance scores for translation (left) and rotation (right).

However, there are several drawbacks to this method. First, the results are not easily interpretable because they consist of two values of different scale. Second, this evaluation method has additional requirements in the form of the availability of the reference map and approximate correspondence of the number of nodes in two maps. The first limitation can not be avoided and there must be a reference map to compare to. Nonetheless, the second limitation could be circumvented by associating nodes retrospectively. It means that after having the corrected map we load both maps into Graph Manipulation Application and note the closest nodes by visual inspection. Next, we are able to measure the distance between specific nodes. This approach was used for Data set 2 because of large difference in the number of graph nodes between evaluated and reference maps (Figure 12).

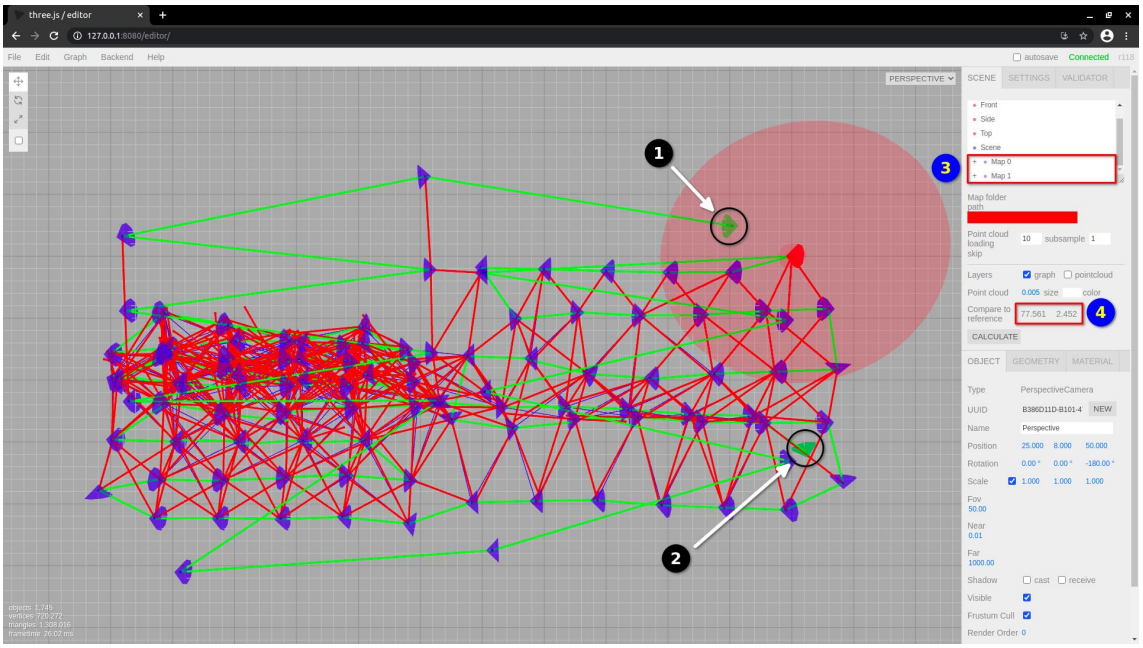

Figure 12: Problem with graph correspondence for measuring distance to a reference map. Map 0 (evaluated) has 91 nodes while Map 1 (reference) has 15 nodes. 1 - Node 14 from Map 1 (reference) and 2 - Node 90 from Map 1 (evaluated) were matched retrospectively. 3 - both maps in the object tree, 4 - calculated distance scores for translation (left) and rotation (right).

5.1.3 Point cloud likelihood

The second alternative to visual inspection does not require a reference map and calculates a relative score, which is useful only when comparing maps of the same environment. This is a useful feature since we compare the results of manipulations on the same map. Nonetheless, this evaluation method has the lowest interpretability of results out of all evaluation methods because it involves map representations and concepts that are not intuitive for an untrained user.

Point cloud likelihood evaluation consists of several steps illustrated on Figure 13. First, we merge both point cloud and NDT submaps into one big map. Next, we iterate through the point cloud and calculate a Mahalanobis [81] distance l from the point to the closest point distributions in neighbor NDT cells (Equation 6).

$$l = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}), \tag{6}$$

where $\mathbf{x}$ is the position vector of the measured point, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are the mean vector and covariance matrix of the NDT cell distribution.

If the distance l is within the threshold of 8, we calculate a negative log-likelihood that a point belongs to the normal distribution we are evaluating using the constant factor term (Equation 7). Otherwise, we assign zero value to the outlier distance,

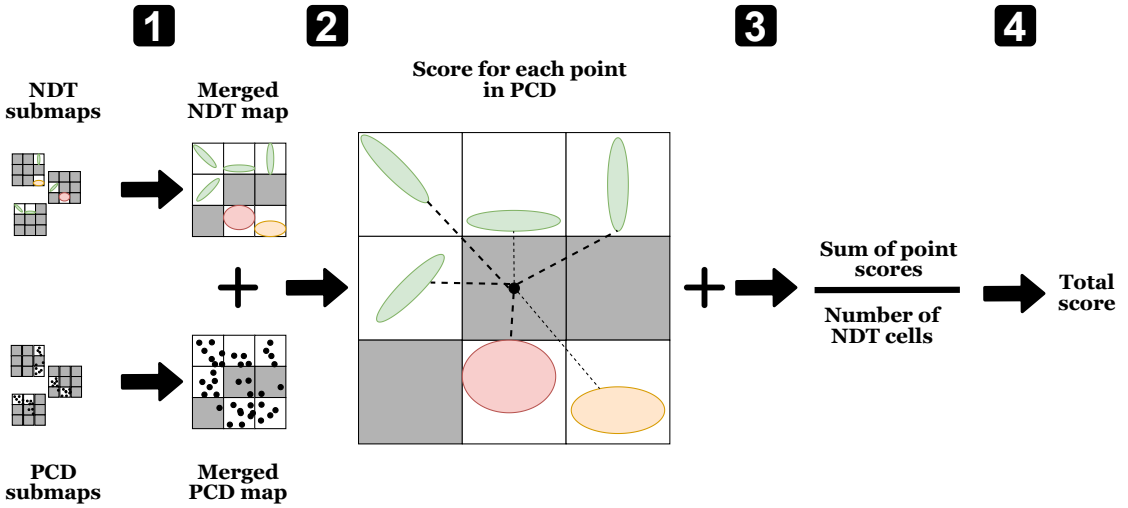

Figure 13: Point cloud likelihood evaluation steps. 1 - merge NDT and PCD submaps into corresponding maps, 2 - calculate likelihood score for each point in the merged PCD map based on the distance to distributions in the merged NDT map while rejecting outliers, 3 - sum up the scores, 4 - divide by number of NDT cells to obtain the final score.

which leads to zero likelihood score.

$$P(\mathbf{x}) = \begin{cases} \frac{1}{(2\pi)^{3/2}\sqrt{\boldsymbol{\Sigma}}} e^{-l/2}, & \text{if } l \leq 8 \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where $P(\mathbf{x})$ is the likelihood of having measured $\mathbf{x}$ given its Mahalanobis distance to the point distribution of the NDT cell. The factor $((2\pi)^{3/2}\sqrt{\boldsymbol{\Sigma}})^{-1}$ [20] scales the point likelihood so that it integrates to one.

The threshold value t equal to 8 was calculated empirically in 2 steps (Equation 8). First, an individual threshold t_{map} was set to 3 standard deviations ($\sigma_{\mathbf{l}}$) from the mean of the distribution of all Mahalanobis distance values $\mathbf{l} = \{l_1, l_2, ..., l_n\}$ for each of 19 evaluated maps. Second, the final threshold value t is the average of all individual threshold values t_{map} rounded to the integer value.

$$\begin{aligned} t_{map} &= 3\sigma_{\mathbf{l}} \\ t &\approx \frac{\sum_{i=1}^{19} t_{map}}{19}. \end{aligned} \tag{8}$$

Finally, we sum up all scores for individual points and normalize by the number of NDT cells in the map, which gives us a final point likelihood score for the map (Equation 9). Lower scores indicate better maps and we are able to compare output maps after different sequences of user manipulations.

$$score = -\frac{\sum_{i=1}^{n} \log(P(\mathbf{x}))}{m}, \tag{9}$$

where n - number of points in the point cloud map, m - number of NDT cells in the NDT map.

Each of the described evaluation methods has its advantages and drawbacks, which are summarized in Table 10. Now that we have several evaluation methods we present experiments and their results in the following sections.

Table 10: Advantages and drawbacks of different evaluation methods

Metric	**Visual**	**Distance**	**Likelihood**
Effort	Low	High	Medium
Objective	No	Yes	Yes
Detect small changes	No	Yes	Yes
Additional requirements	No	Reference map	No
Single value	Yes	No	Yes
Effectiveness	Low	Medium	High
Interpretability	High	Medium	Low

5.2 Data set 1. Challenging outdoor environment

The first experiment was conducted on a challenging outdoor environment data set (Figure 14). One of the main challenges outdoors has to do with unreliable GNSS signal, which results in incorrect measurements and absolute constraints. GNSS signal could be weak, distorted or lost in tunnels, around tall buildings. Another challenge outdoors is trees, which could make it difficult to register submaps.

Graph Manipulation Application was evaluated on a data set recorded in Pasila, Helsinki using an autonomous vehicle. For evaluation purposes, only a part of the recorded trajectory was selected. The length of the evaluated trajectory is around 310m. The main sensor used for this recording is the Velodyne VLP-16 LiDAR. SLAM algorithm generated 61 submaps of size 160m×160m×20m each with distance of 5m between their centers (graph nodes). LiDAR generated point clouds with 50000 to 150000 points for each submap. For each submap except the last, SLAM algorithm created map matching constraints connecting successive submaps. Additionally, it generated 10 false positive loop closure constraints connecting non-successive submaps.

Figure 14 shows the output map of the SLAM algorithm. There are visibly distorted map segments. Because of buildings surrounding the road, the GNSS signal was reflected and incorrectly registered by the receiver. It leads to map segments that are off the correct trajectory.

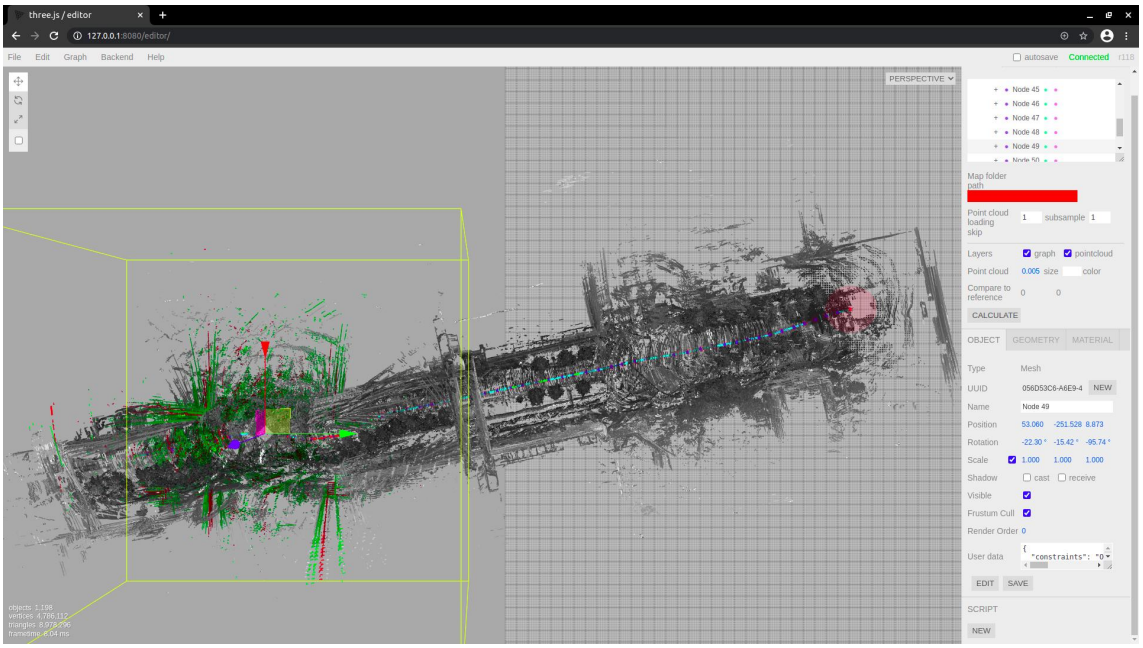

Figure 14: Original distorted map for Data set 1. The main source of distortions are poor GNSS-RTK measurements in an urban environment that includes tall buildings and bridges.

5.2.1 Design

In order to test the effectiveness of Graph Manipulation Application in the described environment, several different manipulations were performed on the input map. The general sequence of actions using the proposed system is the following:

1. Import a broken map;
2. Visually inspect the map to find problematic areas such as distorted graph elements and misaligned point clouds;
3. Manipulate the graph;
4. Run map optimization;
5. Export updated map.

12 different manipulation sequences were chosen by the author to correct the wrong submap poses in the distorted segments. The manipulations were performed during the testing phase by the author. The manipulations are listed in Table 11. A reference map for translation and rotation scores was obtained from the same data using processed RTK-GNSS measurements to smooth the noise. Unfortunately, since there are no validated ground truth pose measurements, the reference map could not be considered as true ground truth map but only as smoothed approximation.

Table 11: Manipulations for Data set 1

ID	Manipulation
OM-1	Remove absolute constraints represented by RTK-GNSS measurements, where they either have bad quality or deviate from the nodes significantly
OM-2	Remove all loop closure constraints in addition to OM-1
OM-3	Remove all map registration constraints in addition to OM-1
OM-4	Remove all loop closure and map registration constraints in addition to OM-1
OM-5	Translate nodes in distorted segments to the approximately correct position in addition to OM-1
OM-6	Translate nodes in distorted segments to the approximately correct position in addition to OM-1 and OM-2
OM-7	Translate nodes in distorted segments to the approximately correct position in addition to OM-1 and OM-3
OM-8	Translate nodes in distorted segments to the approximately correct position in addition to OM-1 and OM-4
OM-9	Rotate nodes in distorted segments to the approximately correct orientation in addition to OM-5
OM-10	Rotate nodes in distorted segments to the approximately correct orientation in addition to OM-6
OM-11	Rotate nodes in distorted segments to the approximately correct orientation in addition to OM-7
OM-12	Rotate nodes in distorted segments to the approximately correct orientation in addition to OM-8

5.2.2 Results

Visual inspection

Figure 15 compares the baseline and corrected maps. The image on the top shows a distorted segment from the baseline map visualized in the Graph Manipulation Application. The selected Node 49 contains a point cloud highlighted in red, while the green colored point clouds belong to neighbor nodes connected by relative constraints. Looking at the colored point clouds, a human user can detect the same environment features such as building silhouettes in different scans and confirm that they are incorrectly positioned. However, the map is corrected on the bottom of Figure 15. The map was corrected with manipulation sequence OM-12 using the Graph Manipulation Application. The map looks better and environment features are aligned so that a human user can identify them.

Table 12 lists manipulation sequences and corresponding results of visual inspection. The best results are obtained after manipulation sequences OM-2 and OM-11. Larger output map images for Data set 1 are presented for reference in Appendix B.

Table 12: Visual inspection results for Data set 1

ID	Result
Baseline	Distorted (Figure B1)
OM-1	Partially corrected (Figure B2)
OM-2	**Corrected** (Figure B3)
OM-3	Partially corrected (Figure B4)
OM-4	Partially corrected (Figure B5)
OM-5	Distorted (Figure B6)
OM-6	Distorted (Figure B7)
OM-7	Distorted (Figure B8)
OM-8	Partially corrected (Figure B9)
OM-9	Partially corrected (Figure B10)
OM-10	Partially corrected (Figure B11)
OM-11	**Corrected** (Figure B12)
OM-12	Partially corrected (Figure B13)

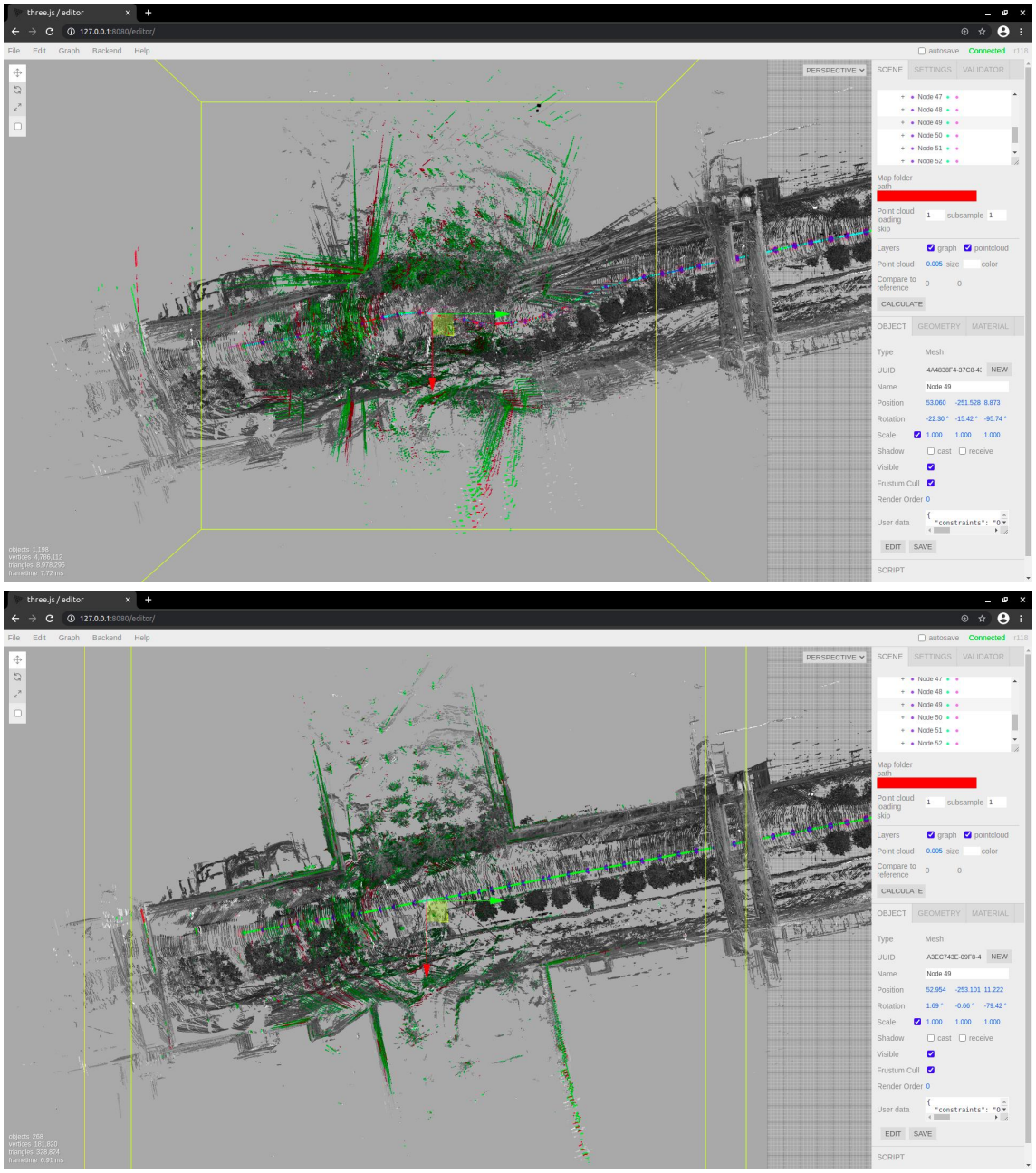

Figure 15: Comparison of the original distorted map segment (top) and the same map segment corrected (bottom) using the Graph Manipulation Application.

Translation and rotation distance

Table 13 shows translation and rotation distance scores of output maps compared to a reference map after each sequence of manipulations. The best scores of 102.489 meters of translation and 1.273 radians of rotation are obtained with three manipulation sequences OM-4, OM-8 and OM-12. Their output map nodes have both the closest translation and rotation distances to the reference map nodes.

Table 13: Translation and rotation distance scores for Data set 1

ID	Translation (m)	Rotation (rad)
Baseline	176.454	15.224
OM-1	109.171	1.647
OM-2	108.733	1.918
OM-3	107.931	1.447
OM-4	**102.489**	**1.273**
OM-5	128.619	4.337
OM-6	104.608	2.518
OM-7	110.939	4.165
OM-8	**102.489**	**1.273**
OM-9	105.871	1.705
OM-10	107.552	1.424
OM-11	104.778	1.318
OM-12	**102.489**	**1.273**

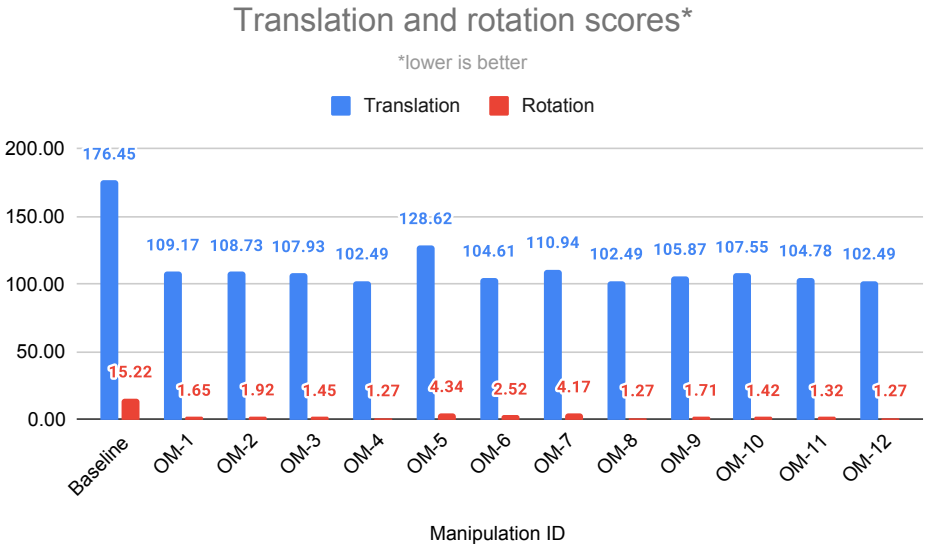

Figure 16: Translation and rotation distance scores for Data set 1.

Point cloud likelihood

Table 14 lists point cloud likelihood scores as well as the corresponding number of outlier points for output maps after each sequence of manipulations. The best score was obtained after manipulation sequence OM-1, where absolute constraints in segments with poor GNSS quality were removed. The lowest number of outlier points was calculated for the baseline map. We note that there is a slight inverse correlation between number of outlier points and map quality score. One explanation could be that worse, non-aligned maps have more dispersed point clouds and therefore more distributions are generated for NDT cells in the corresponding NDT maps. This increases probability of a random point being within the Mahalanobis distance threshold to the closest distribution. Whereas, better aligned maps have more empty NDT cells and greater distances from a random point to the closest distribution as a result.

Table 14: Likelihood scores for Data set 1

ID	Score	Number of outlier points
Baseline	-96.730	**159,473**
OM-1	**-170.017**	170,974
OM-2	-169.803	166,527
OM-3	-160.224	190,130
OM-4	-167.829	191,192
OM-5	-143.164	171,043
OM-6	-143.849	182,817
OM-7	-133.284	190,159
OM-8	-167.829	191,192
OM-9	-160.386	171,556
OM-10	-164.250	175,231
OM-11	-167.592	188,472
OM-12	-167.829	191,192

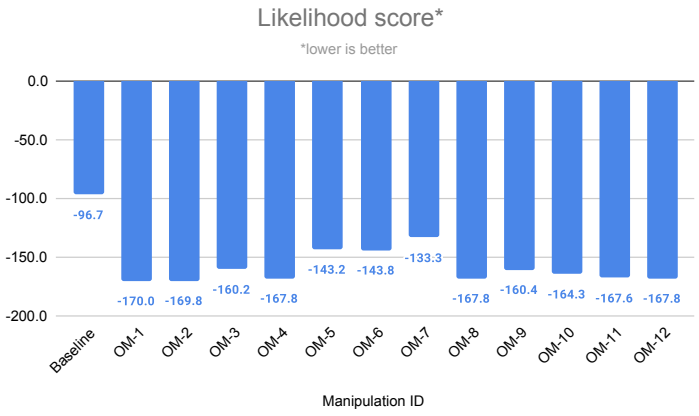

Figure 17: Likelihood scores for Data set 1.

Comparison of results

Table 15 lists all evaluation results for Data set 1.

Table 15: Evaluation results for Data set 1

ID	Visual	Translation (m)	Rotation (rad)	Likelihood
Baseline	Distorted	176.454	15.224	-96.730
OM-1	Partially corrected	109.171	1.647	**-170.017**
OM-2	**Corrected**	108.733	1.918	-169.803
OM-3	Partially corrected	107.931	1.447	-160.224
OM-4	Partially corrected	**102.489**	**1.273**	-167.829
OM-5	Distorted	128.619	4.337	-143.164
OM-6	Distorted	104.608	2.518	-143.849
OM-7	Distorted	110.939	4.165	-133.284
OM-8	Partially corrected	**102.489**	**1.273**	-167.829
OM-9	Partially corrected	105.871	1.705	-160.386
OM-10	Partially corrected	107.552	1.424	-164.250
OM-11	**Corrected**	104.778	1.318	-167.592
OM-12	Partially corrected	**102.489**	**1.273**	-167.829

We notice a disagreement in results between the evaluation methods. If we use only visual inspection method, we would choose manipulation sequences OM-2 and OM-11 as the best. On the other hand, reference map comparisons give us manipulation sequences OM-4, OM-8 and OM-12 as the best. Yet another best result from point cloud likelihood scores indicates manipulation sequence OM-1 as the best. Based on that we can notice that the results are inconclusive. However, they are still useful in practice. Figure 18 demonstrates that Graph Manipulation Application is effective in fixing the distorted map from Data set 1 across all manipulation sequences.

Another interesting finding from Table 15 is that results of visual inspection of type Distorted have the largest rotation distance scores. We could hypothesize that it is easier to detect rotation differences visually and hence the correlation. However we need more data to confirm this.

Nevertheless choosing one evaluation method could help to resolve inconsistencies in results. First of all, we note that the reference map was not the true ground truth map and we need a better reference map in future to make better conclusions. If we discard the results from reference map comparisons, we are left with two evaluation methods and three candidate manipulation sequences OM-1, OM-2 and OM-11. Cross-checking the results we can conclude that manipulation sequence OM-2 gives close to the best results in both evaluation methods. Answering the research question RQ3 for outdoor environments, it seems that the most effective manipulations include removing absolute constraints with poor quality and additionally removing wrong loop closure constraints. It worked in this map because we had GNSS-RTK measurements with good quality both at the beginning and at the end of the trajectory. In general, it is best to remove only false positive loop closures and keep true positive loop

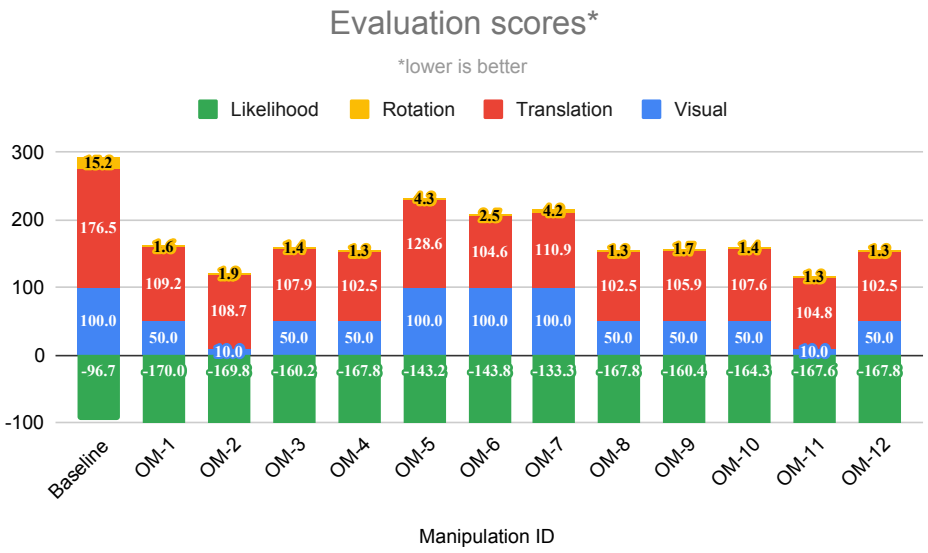

Figure 18: Results for Data set 1.

closures. However, we should note that these particular manipulations could not be generalized without more experiments in different outdoor environments.

5.3 Data set 2. Challenging indoor environment

The second data set for Graph Manipulation Application experiments is a challenging indoor environment with no GNSS signal. The data set was recorded in a warehouse with repetitive rows of shelves, which made it hard for a robot to register loop closures correctly. The length of the evaluated trajectory is around 460m. The primary sensor was RoboSense RS-16 LiDAR. SLAM algorithm generated 91 submaps of size 120m×120m×10m each with distance of 5m between their centers (graph nodes). LiDAR generated point clouds with 250000 to 1000000 points for each submap. For each submap except the last, SLAM algorithm created map matching constraints connecting successive submaps. Additionally, it generated 434 mostly false positive loop closure constraints connecting non-successive submaps (Figure 19), which led to the SLAM failure.

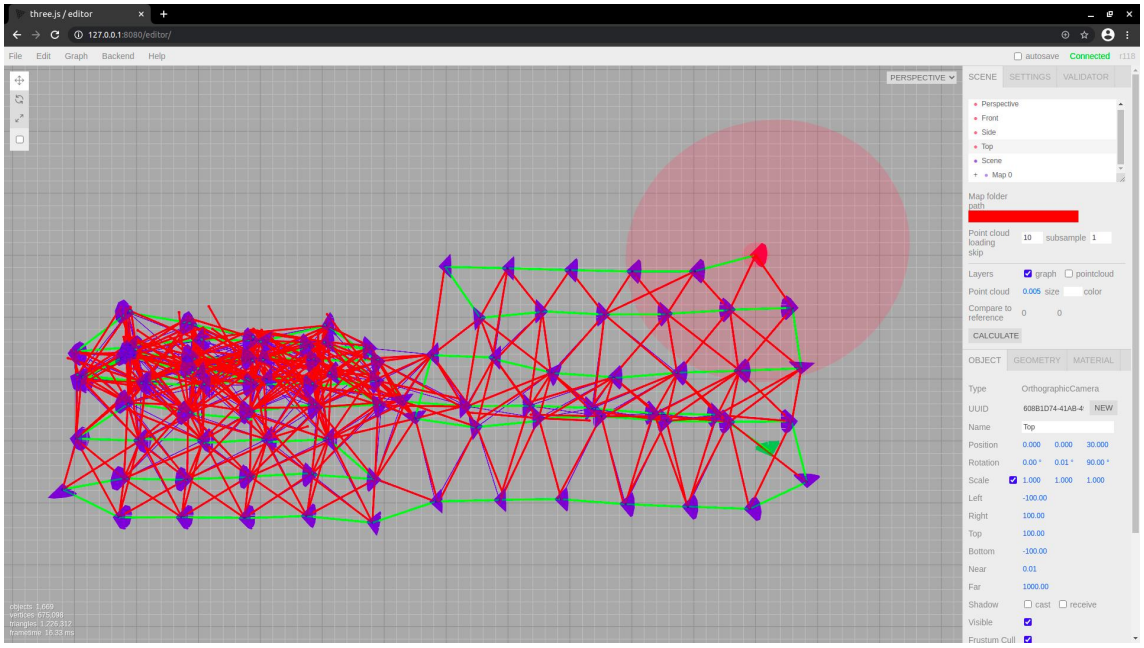

Figure 19: Graph for the original distorted map of Data set 2. The main source of distortions are false positive loop closures created because of perceptual aliasing in repetitive rows of shelves. Loop closures are indicated as red edges connecting graph nodes.

5.3.1 Design

In order to test the effectiveness of Graph Manipulation Application in the described environment, several manipulations were performed on the input map. The general sequence of actions using the proposed system is the following:

1. Import a broken map;
2. Visually inspect the map to find problematic areas such as distorted graph elements and misaligned point clouds;
3. Manipulate the graph;

4. Run map optimization;
5. Repeat until satisfied with map quality;
6. Export updated map.

5 different manipulation sequences listed in Table 16 were selected and tested by the author.

Table 16: Manipulations for Data set 2

ID	Manipulation
IM-1	Remove all relative constraints represented by loop closures
IM-2	Add 3 new loop closure constraints in addition to the manipulation IM-1
IM-3	Add 6 new loop closure constraints in addition to the manipulation IM-1
IM-4	Add 10 new loop closure constraints in addition to the manipulation IM-1
IM-5	Add 19 new loop closure constraints in addition to the manipulation IM-1

A reference map for translation and rotation scores was obtained from the same data using larger submap distances, which helped to converge to a good global estimate. Unfortunately, since there are no globally estimated and validated poses, the reference map could not be considered as true ground truth map but only as an approximation.

5.3.2 Results

Visual inspection

Figure 20 compares the baseline and final corrected maps. The image on the top shows the distorted baseline map visualized in the Graph Manipulation Application. The map on the bottom is corrected iteratively using the Graph Manipulation Application until the output map is visually consistent. The corrected map contains distinguishable rows of shelves while the original map does not.

Table 17 lists manipulation sequences and corresponding results of visual inspection. The best results are obtained after manipulation sequences IM-4 and IM-5. Larger output map images for Data set 2 are presented for reference in Appendix C.

Table 17: Visual inspection results for Data set 2

ID	Result
Baseline	Distorted (Figure C1)
IM-1	Partially corrected (Figure C2)
IM-2	Partially corrected (Figure C3)
IM-3	Partially corrected (Figure C4)
IM-4	**Corrected** (Figure C5)
IM-5	**Corrected** (Figure C6)

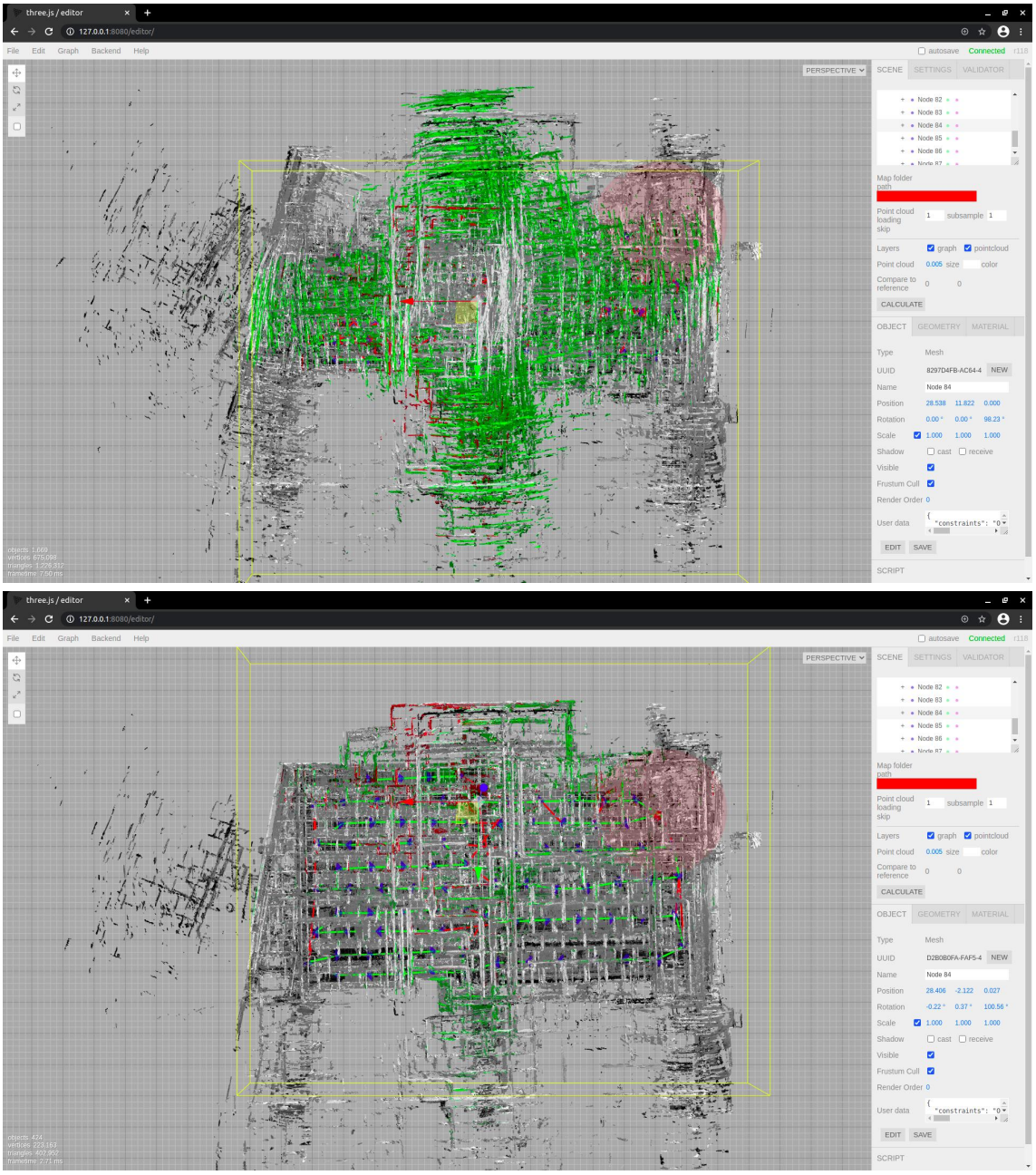

Figure 20: Comparison of the original distorted map (top) and the same map corrected (bottom) using the Graph Manipulation Application. Both translation and rotation errors were corrected.

Translation and rotation distance

Evaluating Data set 2 with distance scores was challenging because the reference map has several times less graph nodes and looking for the closest node on different maps would produce different results. The solution in this case was to compare the reference map and visually best corrected candidate map and find node correspondences manually. Then the next step is to compare the same corresponding nodes on different maps.

Table 18 demonstrates translation and rotation distance scores of output maps compared to a reference map after each sequence of manipulations. The best translation score of 18.664 meters from the reference map was obtained with manipulation sequence IM-3, i.e. removing all loop closures first and adding 6 new correct loop closures. The best rotation score of 1.665 radians was obtained with manipulation sequence IM-4, i.e. removing all loop closures and adding 10 new correct loop closures.

Table 18: Translation and rotation distance scores for Data set 2

ID	Translation (m)	Rotation (rad)
Baseline	77.561	2.452
IM-1	34.83	2.441
IM-2	22.822	1.937
IM-3	**18.664**	1.751
IM-4	20.196	**1.665**
IM-5	19.2	1.666

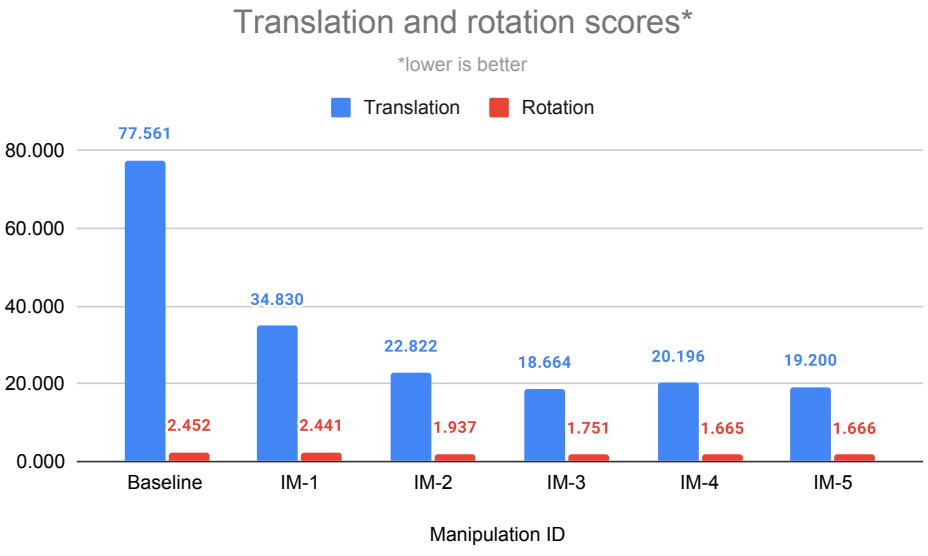

Figure 21: Translation and rotation distance scores for Data set 2.

Point cloud likelihood

Table 19 lists point cloud likelihood scores as well as the corresponding number of outlier points for output maps after each sequence of manipulations. The best score is obtained after manipulation sequence IM-5, i.e. removing all loop closures and adding 19 new loop closures. The lowest number of outlier points was calculated in the baseline map.

Table 19: Likelihood scores for Data set 2

ID	Score	Number of outlier points
Baseline	-812.748	**923,213**
IM-1	-949.520	1,068,751
IM-2	-1,033.380	1,046,213
IM-3	-1,139.150	1,035,255
IM-4	-1,182.010	1,036,212
IM-5	**-1,188.330**	1,038,867

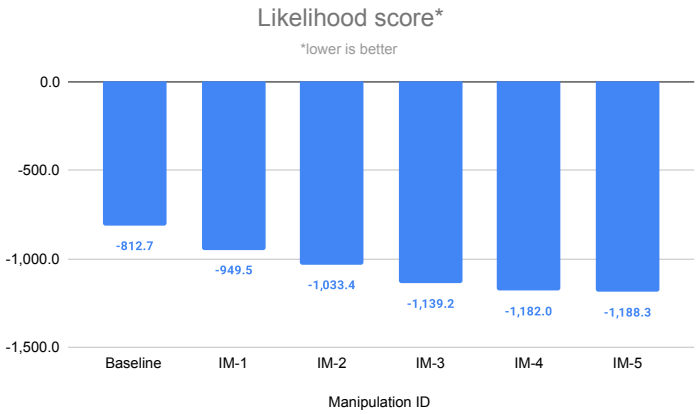

Figure 22: Likelihood scores for Data set 2.

Comparison of results

Fugure 23 and Table 20 present combined evaluation results for Data set 2.

Table 20: Evaluation results for the Data set 2

ID	Visual	Translation (m)	Rotation (rad)	Likelihood
Baseline	Distorted	77.561	2.452	-812.748
IM-1	Partially corrected	34.83	2.441	-949.520
IM-2	Partially corrected	22.822	1.937	-1,033.380
IM-3	Partially corrected	**18.664**	1.751	-1,139.150
IM-4	**Corrected**	20.196	**1.665**	-1,182.010
IM-5	**Corrected**	19.2	1.666	**-1,188.330**

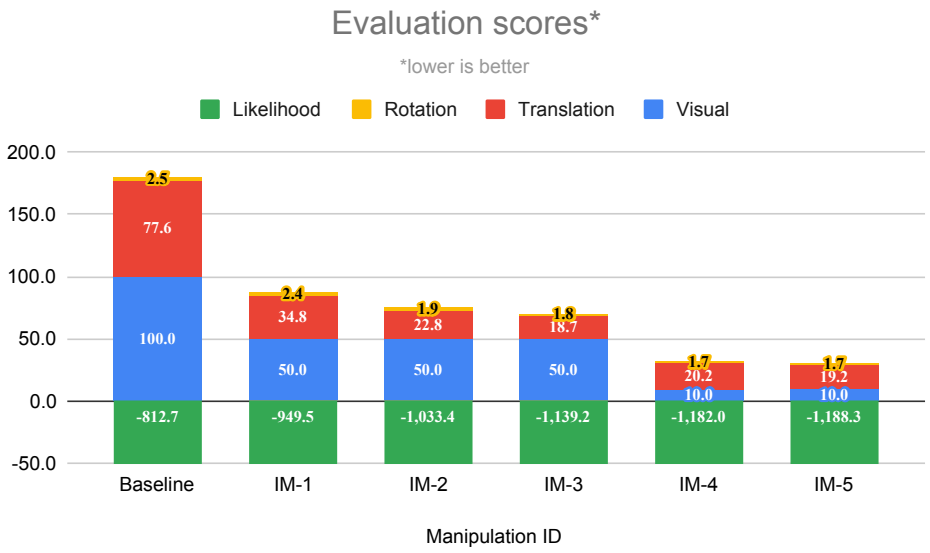

Figure 23: Results for Data set 2.

There is less disagreement in results for Data set 2 and the best candidate manipulation sequences IM-4 and IM-5 include removing all initial loop closure constraints and iterative addition of 10 and 19 new loop closure constraints correspondingly. Manipulation IM-1 was needed in this case because there were too many false positive loop closures to remove them manually. Additionally, Figure 23 shows that Graph Manipulation Application is effective in fixing the distorted map from Data set 2 across all manipulation sequences as well.

Answering the research question RQ3 for indoor environments, we can conclude that removing false positive loop closures and iteratively adding more true positive loop closures helps to converge to the best solution.

5.4 Tests against requirements

Table 21 lists tests conducted to verify that Graph Manipulation Application satisfies the system requirements defined in Chapter 3. System requirement R-6 was taken into account during the system design but was not tested on platforms other than Linux. However, since Graph Manipulation Application is a web application, it should be available for use on other platforms as well. System requirements R-18 and R-21 were not implemented for this thesis since they do not affect system performance.

Table 21: Tests against the requirements

ID	Test	Status
R-1	Discover map errors visually	Passed
R-2	Fix indoor map	Passed
R-2	Fix outdoor map	Passed
R-3	Confirm that the system is used in offline SLAM mode	Passed
R-4	Confirm that the system uses the existing map format	Passed
R-5	Confirm that the system architecture is modular	Passed
R-6	Confirm that the system is platform-independent	Not tested
R-7	Confirm that the system uses existing communication libraries	Passed
R-8	Confirm that the system uses the existing storage type	Passed
R-9	Confirm that the system uses existing graph optimization library	Passed
R-10	Confirm that the system uses existing submap registration algorithm/library	Passed
R-11	Load existing map files	Passed
R-12	Close currently opened map	Passed
R-13	Save currently opened map	Passed
R-14	Confirm graph components visualized correctly	Passed
R-15	Confirm point clouds visualized correctly	Passed
R-16	Switch on/off layers for graph components and point clouds	Passed
R-17	Select an object and verify that its attributes are displayed	Passed
R-18	Move and rotate graph nodes to see them changed	Not implemented
R-18	Run map optimization to see objects change their poses	Not implemented
R-19	Change point cloud's point size	Passed
R-20	Change point cloud's point color	Passed
R-21	Confirm mapping errors visualized correctly	Not implemented
R-22	Move a graph node	Passed
R-23	Remove a graph node	Passed
R-24	Add a new relative constraint between two nodes	Passed
R-25	Change covariance values on an existing relative constraint	Passed
R-26	Remove an existing relative constraint	Passed
R-27	Confirm a map origin visualized correctly	Passed
R-28	Confirm a map trajectory end visualized correctly	Passed
R-29	Move a graph node with a map matching constraint	Passed
R-30	Create a new relative constraint and register it manually	Passed
R-31	Run graph optimization	Passed
R-32	Confirm that the system notifies of back-end service failures	Passed

5.5 Summary

The results from Chapter 5 demonstrate that even the least effective manipulations improve the map quality compared to the baseline map. However, some manipulations result in better scores and require less effort. The challenge is to choose the most effective workflows even if the results using different methods differ. The basic framework for choosing the best map is to inspect output maps visually first, then calculate distance scores if a verified reference map is available and finally calculate point cloud likelihood scores if the reference map is not available or could not be verified.

Using the results presented in this chapter, we can answer the research question RQ2 posed in Chapter 1. The proposed Graph Manipulation Application is effective in fixing the distorted maps from both data sets. The most obvious way to confirm this is to compare the map segments from Figure 15 for the outdoor data set and Figure 20 for the indoor data set. Additional confirmation of results is presented in Table 15 and Figure 18 for Data set 1 and in Table 20 and Figure 23 for Data set 2, where most of the manipulations in the proposed software application resulted in better scores.

6 Conclusion and future work

This thesis presented a Graph Manipulation Application that was designed to correct algorithmic SLAM failures with the help of a human expert by editing the graph underlying the environment map. The main conclusion of this work is that the proposed system is effective at correcting the environment map after SLAM fails to converge to a globally consistent map estimate. The main research problem was decomposed into three research questions. The first research question RQ1 asked for the main requirements for the design and development of the solution. This question was answered in Chapter 3 by compiling a list of main requirements in Table 7. The second research question RQ2 asked if the proposed solution is effective in correcting a distorted map. The results from Chapter 5 confirm the effectiveness of the proposed solution. The third research question RQ3 asked for the most effective sequences of actions to correct a distorted map using the solution. Section 5.5 proposed a basic framework for modifying the graph structure based on the results of experiments conducted for this work. However, research question RQ3 was not fully answered based on the mixed results for Data set 1 and requires conducting more experiments and verifying the results based on the evaluation methods presented in Chapter 5.

There are several directions to explore for the future development of Graph Manipulation Application. First, the functionality could be extended to include other stages of the mapping process. For example, one potentially useful feature would be to allow merging two separate maps into one. Often times robots have overlapping maps of the same environment collected at different times and using different parameters. However, they represent the same environment and merging their underlying graphs would increase map size as well as could potentially increase graph consistency with more available nodes and constraints. Another direction to explore for future development is including into Graph Manipulation Application prior maps of the environment. For outdoor environments, API services such as Open Street Map are available that could be used to ground the map obtained by a robot to the external context. For indoor maps, building plans and design files are usually available that could be used for the same purpose. Finally, another feature to consider is to automate some of the manipulations by introducing templates and presets to align and group objects.

The *robust-perception age* of SLAM research [4] focuses on two of the biggest issues of robust performance and scalability of long-term robot operation in varying environments. The proposed Graph Manipulation Application can potentially help with both problems. The first issue is addressed by users effectively correcting and extending 3D environment maps necessary for robust localization performance. The second issue could be solved by users helping different robots to share maps among each other and reuse the existing man-made maps in order to quickly scale the range of robot operation. However, future work is needed to extend features and verify results with various challenging environments. Ironically, this thesis demonstrated that if we ever want to reach the dream of autonomous mobile robots helping us and navigating safely and reliably around us, we might as well help them first.

References

[1] S. Banker, *The autonomous mobile robot market is taking off like a rocket ship*, Forbes, https://www.forbes.com/sites/stevebanker/2019/03/11/the-autonomous-mobile-robot-market-is-taking-off-like-a-rocket-ship/, accessed: 28/11/2020.

[2] *Automated vehicles for safety*, NHTSA, https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety, accessed: 28/11/2020.

[3] U. Frese, "Interview: Is SLAM solved?" *KI - Künstliche Intelligenz*, vol. 24, no. 3, pp. 255–257, 2010.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. Leonard, "Past, present, and future of simultaneous localization and mapping: Towards the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[5] N. Sünderhauf, "Robust optimization for simultaneous localization and mapping," Ph.D. dissertation, Technischen Universitat Chemnitz, 2012.

[6] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (SLAM) part I: The essential algorithms," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[7] T. Bailey and H. Durrant-Whyte, "Simultaneous localisation and mapping (SLAM) part II: State of the art," *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[8] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer, 2016.

[9] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe, "A review of recent developments in simultaneous localization and mapping," in *6th International Conference on Industrial and Information Systems (ICIIS 2011)*, IEEE, 2011, pp. 477–482.

[10] M. Kaess and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Transactions on Robotics*, p. 14, 2008.

[11] A. Broggi, P. Grisleri, and P. Zani, "Sensors technologies for intelligent vehicles perception systems: A comparison between vision and 3D-LIDAR," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, IEEE, 2013, pp. 887–892.

[12] H. Choset, *Localization, mapping, SLAM and the Kalman filter according to George*, https://www.cs.cmu.edu/~motionplanning/lecture/Chap8-Kalman-Mapping_howie.pdf, accessed: 28/11/2020.

[13] H. Moravec and A. E. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of the 1985 IEEE International Conference on Robotics and Automation*, 1985, pp. 116–121.

[14] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proceedings of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[15] P. Biber and W. Straßer, "The normal distributions transform: A new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(Cat. No. 03CH37453)*, IEEE, vol. 3, 2003, pp. 2743–2748.

[16] J. Saarinen, H. Andreasson, T. Stoyanov, J. Ala-Luhtala, and A. J. Lilienthal, "Normal distributions transform occupancy maps: Application to large-scale online 3D mapping," in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2013, pp. 2233–2238.

[17] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based SLAM," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010.

[18] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, 2001.

[19] T. Stoyanov, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1377–1393, 2012.

[20] M. Magnusson, "The three-dimensional normal-distributions transform: An efficient representation for registration, surface analysis, and loop detection," Ph.D. dissertation, Örebro universitet, 2009.

[21] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the 3rd International Conference on 3D Digital Imaging and Modeling (3DIM)*, IEEE, 2001, pp. 145–152.

[22] *General graph optimization (g^2o)*, https://github.com/RainerKuemmerle/g2o, accessed: 21/12/2020.

[23] *iSAM: Incremental smoothing and mapping*, http://people.csail.mit.edu/kaess/isam/, accessed: 21/12/2020.

[24] *Georgia Tech smoothing and mapping library (GTSAM)*, https://github.com/borglab/gtsam, accessed: 21/12/2020.

[25] T. L. Dean and K. Kanazawa, "Probabilistic temporal reasoning.," in *AAAI*, 1988, pp. 524–529.

[26] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, no. 3, pp. 52–57, 2002.

[27] J. Diebel and S. Thrun, "An application of Markov random fields to range sensing," *Advances in Neural Information Processing Systems*, vol. 18, pp. 291–298, 2005.

[28] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g^{2}o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2011, pp. 3607–3613.

[29] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient sparse pose adjustment for 2d mapping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2010, pp. 22–29.

[30] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the Bayes tree," *The International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012.

[31] M. Milford, D. Prasser, and G. Wyeth, "Experience mapping: Producing spatially continuous environment representations using RatSLAM," in *Proceedings of the 2005 Australasian Conference on Robotics and Automation*, Australian Robotics and Automation Association Inc, 2005.

[32] T. Duckett, S. Marsland, and J. Shapiro, "Fast, on-line learning of globally consistent maps," *Autonomous Robots*, vol. 12, no. 3, pp. 287–300, 2002.

[33] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localization and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 196–207, 2005.

[34] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[35] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Transactions on Robotics*, vol. 22, no. 6, pp. 1100–1114, 2006.

[36] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2006, pp. 2262–2269.

[37] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard, "A tree parameterization for efficiently computing maximum likelihood maps using gradient descent.," in *Robotics: Science and Systems*, vol. 3, 2007, p. 9.

[38] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, and A. J. Davison, "Slam++: Simultaneous localisation and mapping at the level of objects," in *Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 1352–1359.

[39] S. S. Puligilla, S. Tourani, T. Vaidya, U. S. Parihar, R. K. Sarvadevabhatla, and K. M. Krishna, "Topological mapping for Manhattan-like repetitive environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 6268–6274.

[40] A. Kelly, *Mobile robotics: mathematics, models, and methods*. Cambridge University Press, 2013.

[41] S. Lowry, N. Sünderhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford, "Visual place recognition: A survey," *IEEE Transactions on Robotics*, vol. 32, no. 1, pp. 1–19, 2015.

[42] G. D. Tipaldi, M. Braun, and K. O. Arras, "FLIRT: Interest Regions for 2D Range Data with Applications to Robot Navigation," in *Experimental Robotics*, O. Khatib, V. Kumar, and G. Sukhatme, Eds., vol. 79, Series Title: Springer Tracts in Advanced Robotics, Springer Berlin Heidelberg, 2014, pp. 695–710.

[43] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[44] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV 2003)*, IEEE Computer Society, vol. 3, 2003, pp. 1470–1470.

[45] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 2, 2006, pp. 2161–2168.

[46] M. J. Milford and G. F. Wyeth, "SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2012, pp. 1643–1649.

[47] W. Churchill and P. Newman, "Experience-based navigation for long-term localisation," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1645–1661, 2013.

[48] K. L. Ho and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance," *Robotics and Autonomous Systems*, vol. 54, no. 9, pp. 740–749, 2006.

[49] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[50] F. Dayoub, G. Cielniak, and T. Duckett, "Long-term experiments with an adaptive spherical view representation for navigation in changing environments," *Robotics and Autonomous Systems*, vol. 59, no. 5, pp. 285–295, 2011.

[51] T. Krajnıék, J. P. Fentanes, O. M. Mozos, T. Duckett, J. Ekekrantz, and M. Hanheide, "Long-term topological localisation for service robots in dynamic environments using spectral maps," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2014, pp. 4537–4542.

[52] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time for pose graph SLAM," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.

[53] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *The International Journal of Robotics Research*, vol. 32, no. 7, pp. 826–840, 2013.

[54] N. Sünderhauf and P. Protzel, "Towards a robust back-end for pose graph SLAM," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2012, pp. 1254–1261.

[55] D. Sabatta, D. Scaramuzza, and R. Siegwart, "Improved appearance-based matching in similar and dynamic environments using a vocabulary tree," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2010, pp. 1008–1013.

[56] E. Olson, J. Strom, R. Goeddel, R. Morton, P. Ranganathan, and A. Richardson, "Exploration and mapping with autonomous robot teams," *Communications of the ACM*, vol. 56, no. 3, pp. 62–70, 2013.

[57] R. Reid and T. Bräunl, "Large-scale multi-robot mapping in MAGIC 2010," in *5th IEEE International Conference on Robotics, Automation and Mechatronics (RAM 2011)*, IEEE, 2011, pp. 239–244.

[58] A. Boeing, M. Boulton, T. Bräunl, B. Frisch, S. Lopes, A. Morgan, F. Ophelders, S. Pangeni, R. Reid, K. Vinsen, N. Garel, C. S. Lee, M. Masek, A. Attwood, M. Fazio, and A. Gandossi, "WAMbot: Team MAGICian's entry to the multi autonomous ground-robotic international challenge 2010," *Journal of Field Robotics*, vol. 29, no. 5, pp. 707–728, 2012.

[59] A. Sidaoui, I. H. Elhajj, and D. Asmar, "Human-in-the-loop augmented mapping," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 3190–3195.

[60] S. Nashed and J. Biswas, "Human-in-the-loop SLAM," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[61] R. Milijaš, J. Oršulić, and S. Bogdan, "When measurements fail: Using an interactive SLAM solution to fight bad odometry," in *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, IEEE, 2020, pp. 1–6.

[62] K. Koide, J. Miura, M. Yokozuka, S. Oishi, and A. Banno, "Interactive 3D graph SLAM for map correction," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 40–47, 2020.

[63] A. Finn, A. Jacoff, M. Del Rose, B. Kania, J. Overholt, U. Silva, and J. Bornstein, "Evaluating autonomous ground-robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 689–706, 2012.

[64] A. Lacaze, K. Murphy, M. Del Giorno, and K. Corley, "Reconnaissance and autonomy for small robots (RASR) team: MAGIC 2010 challenge," *Journal of Field Robotics*, vol. 29, no. 5, pp. 729–744, 2012.

[65] J. Butzke, K. Daniilidis, A. Kushleyev, D. D. Lee, M. Likhachev, C. Phillips, and M. Phillips, "The University of Pennsylvania MAGIC 2010 multi-robot unmanned vehicle system," *Journal of Field Robotics*, vol. 29, no. 5, pp. 745–761, 2012.

[66] E. Olson, J. Strom, R. Morton, A. Richardson, P. Ranganathan, R. Goeddel, M. Bulic, J. Crossman, and B. Marinier, "Progress toward multi-robot reconnaissance and the MAGIC 2010 competition," *Journal of Field Robotics*, vol. 29, no. 5, pp. 762–792, 2012.

[67] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[68] W. Burgard, M. Ruhnke, and B. Steder, *Techniques for 3D mapping*, http://ais.informatik.uni-freiburg.de/teaching/ss16/robotics/slides/17-3dmapping.pdf, accessed: 21/12/2020.

[69] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, "Survey of robot 3D path planning algorithms," *Journal of Control Science and Engineering*, vol. 2016, 2016.

[70] S. Agarwal, K. Mierle, *et al.*, *Ceres solver*, http://ceres-solver.org.

[71] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 1271–1278.

[72] *ISO/IEC/IEEE 29148:2018 Systems and software engineering — Life cycle processes — Requirements engineering*, ISO, https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/20/72089.html, accessed: 28/11/2020.

[73] *Systems Engineering Handbook*, NASA, http://www.nasa.gov/seh/index.html, accessed: 28/11/2020.

[74] M. Cohn, *User Stories and User Story Examples by Mike Cohn*, Mountain Goat Software, https://www.mountaingoatsoftware.com/agile/user-stories, accessed: 21/12/2020.

[75] *The C4 model for visualising software architecture*, https://c4model.com/, accessed: 21/12/2020.

[76] *ROS.org | Powering the world's robots*, https://www.ros.org/, accessed: 21/12/2020.

[77] *rosbridge_suite - ROS Wiki*, https://wiki.ros.org/rosbridge_suite, accessed: 21/12/2020.

[78] *Three.js – JavaScript 3D library*, https://threejs.org/, accessed: 21/12/2020.

[79] *rosbridge_server - ROS Wiki*, https://wiki.ros.org/rosbridge_server, accessed: 21/12/2020.

[80] *rosbridge_library - ROS Wiki*, https://wiki.ros.org/rosbridge_library, accessed: 21/12/2020.

[81] P. C. Mahalanobis, "On the generalized distance in statistics," National Institute of Science of India, 1936.

A Extended set of stakeholders

Table A1: The extended set of stakeholders

ID	Type	Role	Perspective	Expectations
S-1	User	External Customer, Robotics Engineer, Autonomous Vehicle Engineer	Primary user	Convenient and fast map correcting process for both indoor and outdoor environments
S-2	Acquirer	External Customer, Chief Technology Officer	Software stack	The best means to build the map
S-3	Developer	Software Engineer	Software development	Fast and modular software development
S-4	Operator	Robot Operator, Autonomous Vehicle Operator	Secondary user	Trustworthy maps, which do not get in the way of the main robot function of navigation
S-5	Maintainer	Software Engineer, Product Owner, Project Manager	Software maintenance	No software bugs, easy to extend
S-6	Regulatory Authority	State and local authorities, Insurers, Traffic planning	External regulation	Robots are safe and reliable. All errors are traceable to the original source

B Data set 1. Visual inspection results

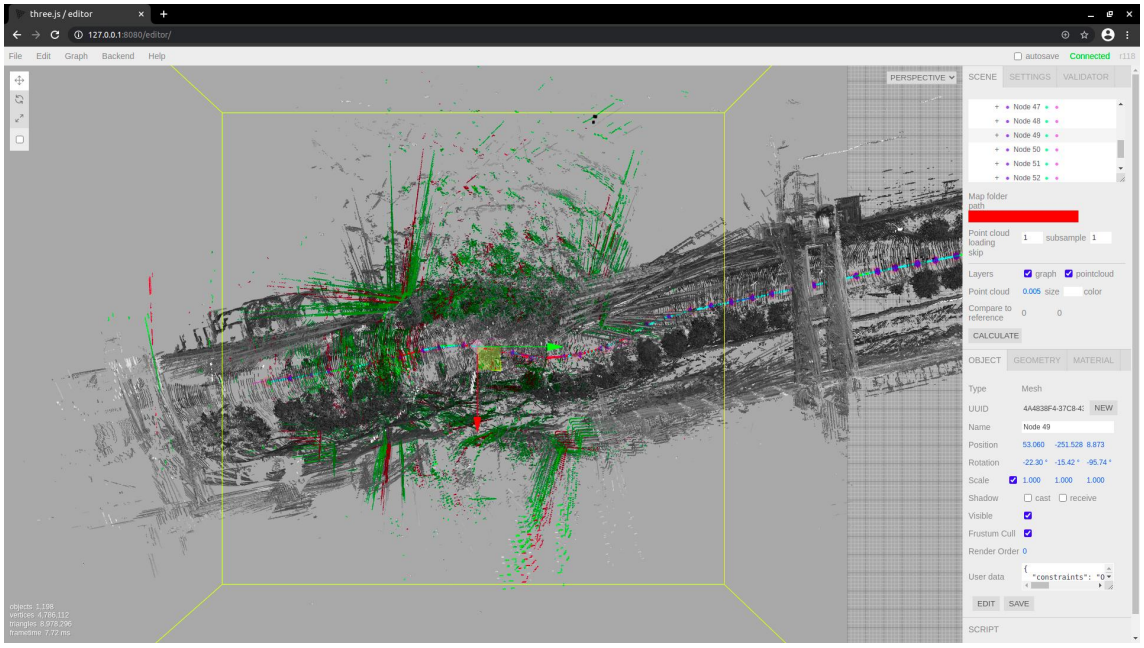

Figure B1: Original distorted map for Data set 1.

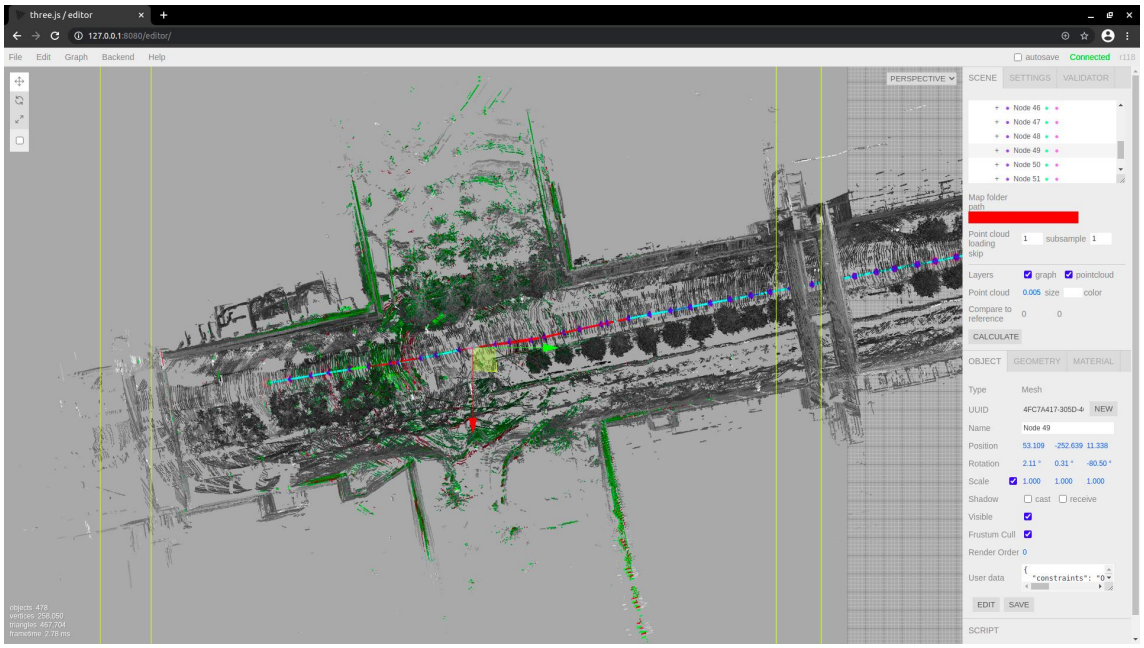

Figure B2: Output map for manipulation sequence OM-1.

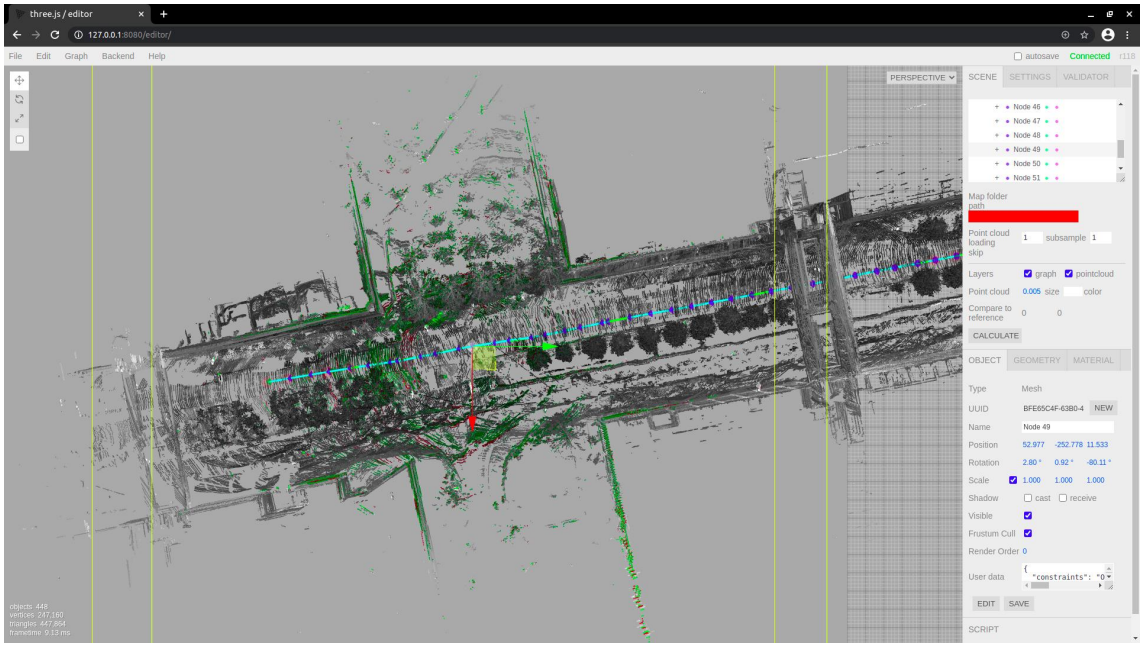

Figure B3: Output map for manipulation sequence OM-2.

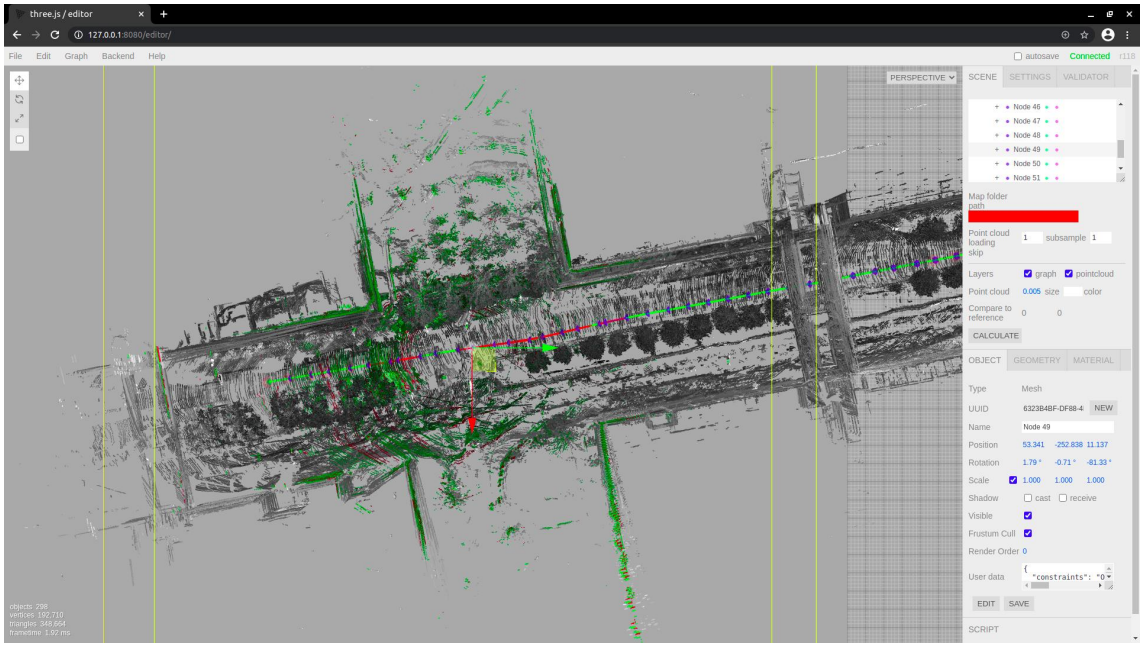

Figure B4: Output map for manipulation sequence OM-3.

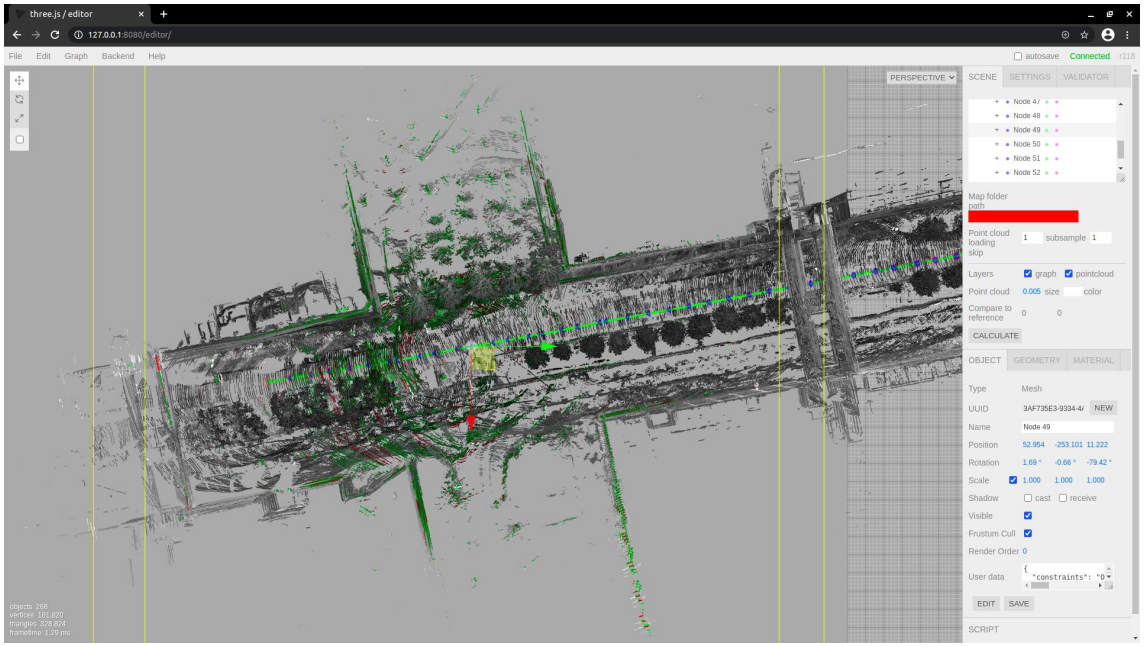

Figure B5: Output map for manipulation sequence OM-4.

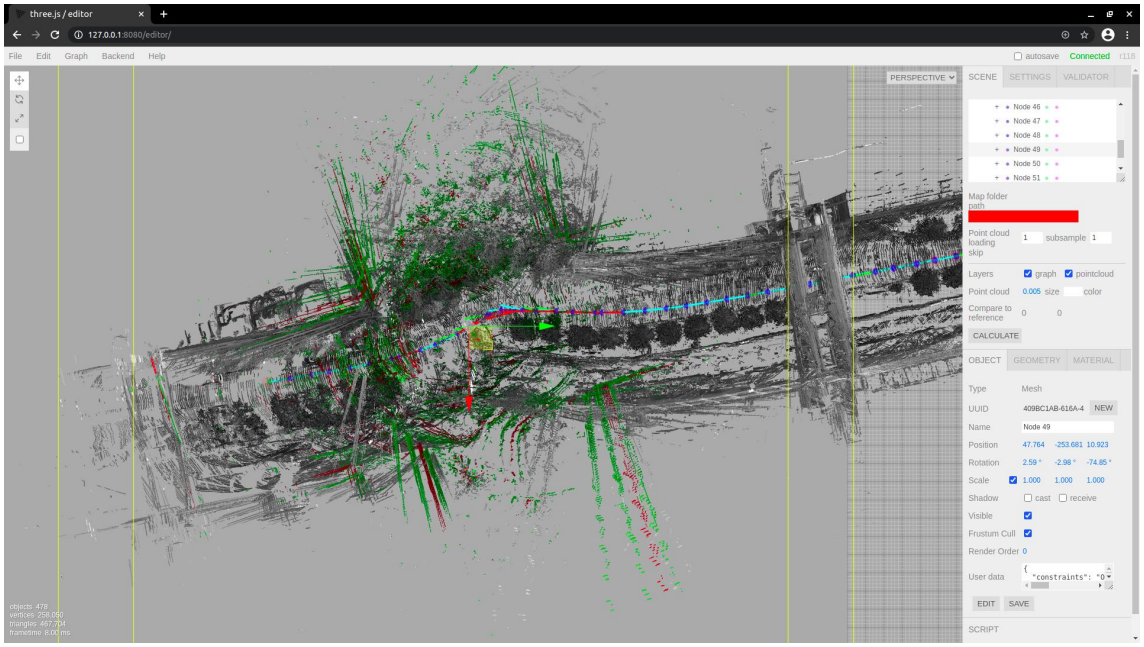

Figure B6: Output map for manipulation sequence OM-5.

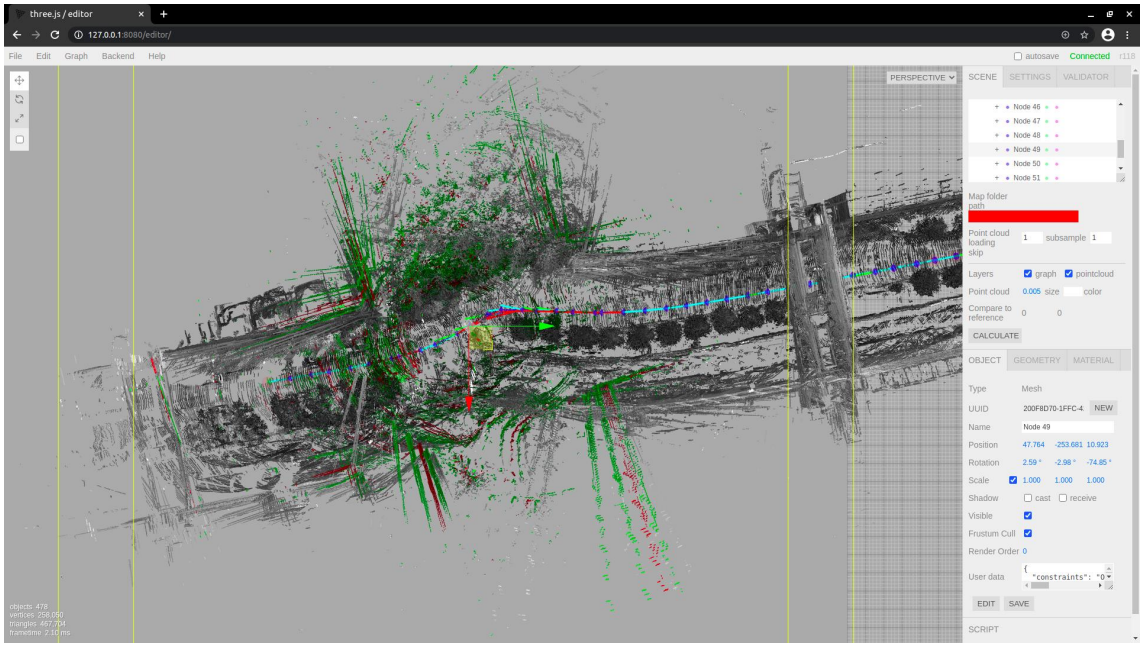

Figure B7: Output map for manipulation sequence OM-6.

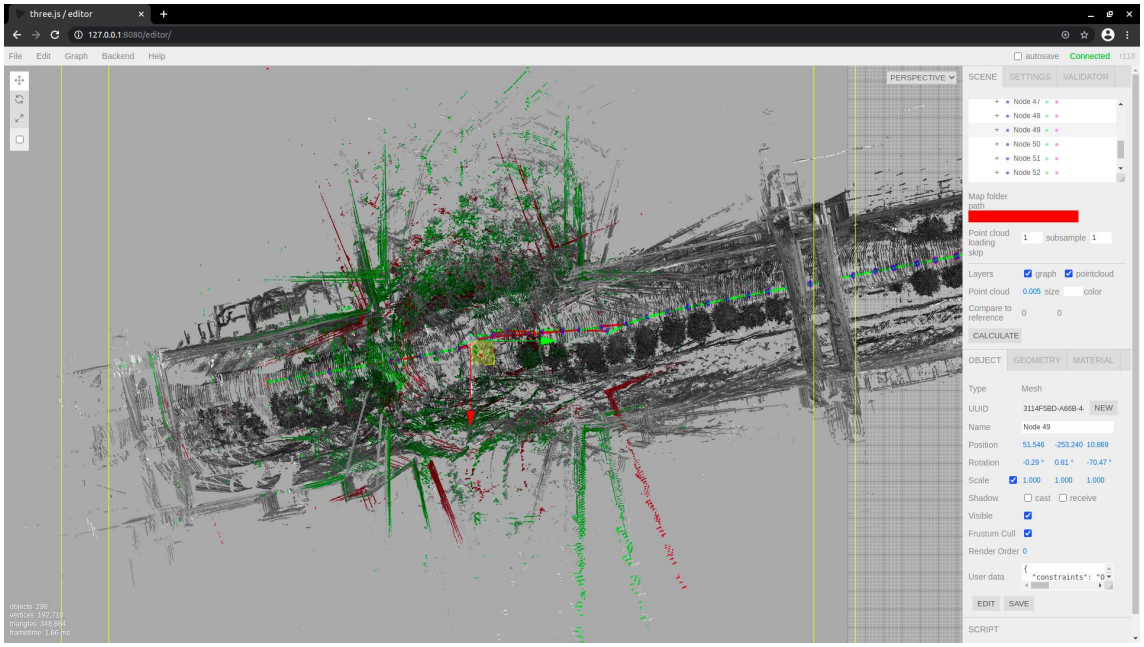

Figure B8: Output map for manipulation sequence OM-7.

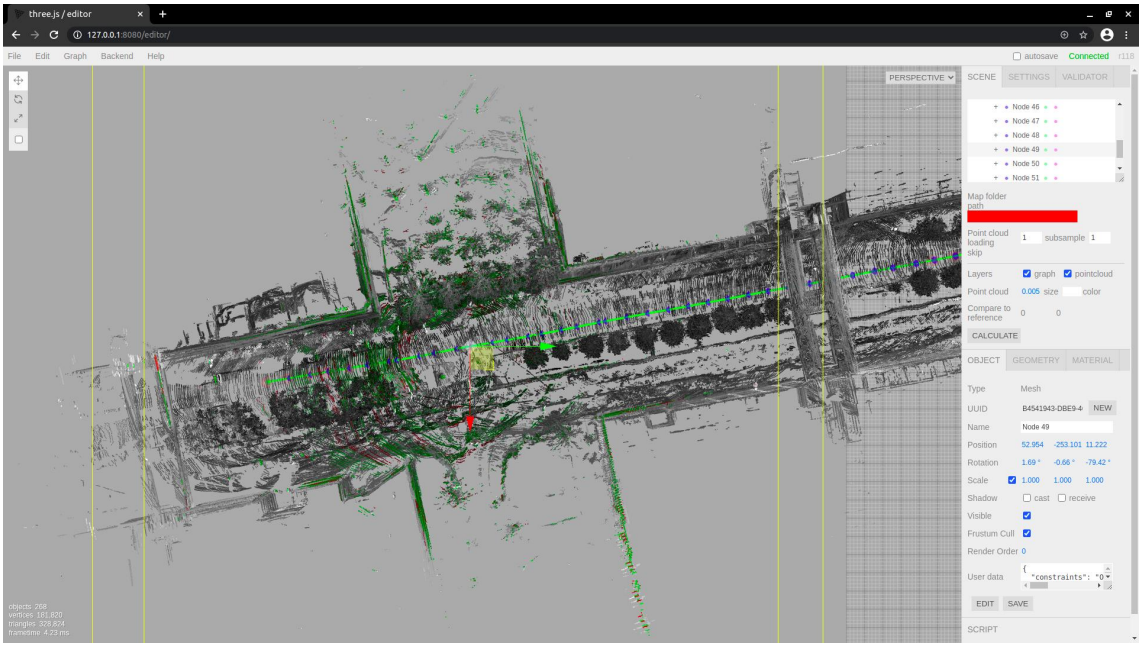

Figure B9: Output map for manipulation sequence OM-8.

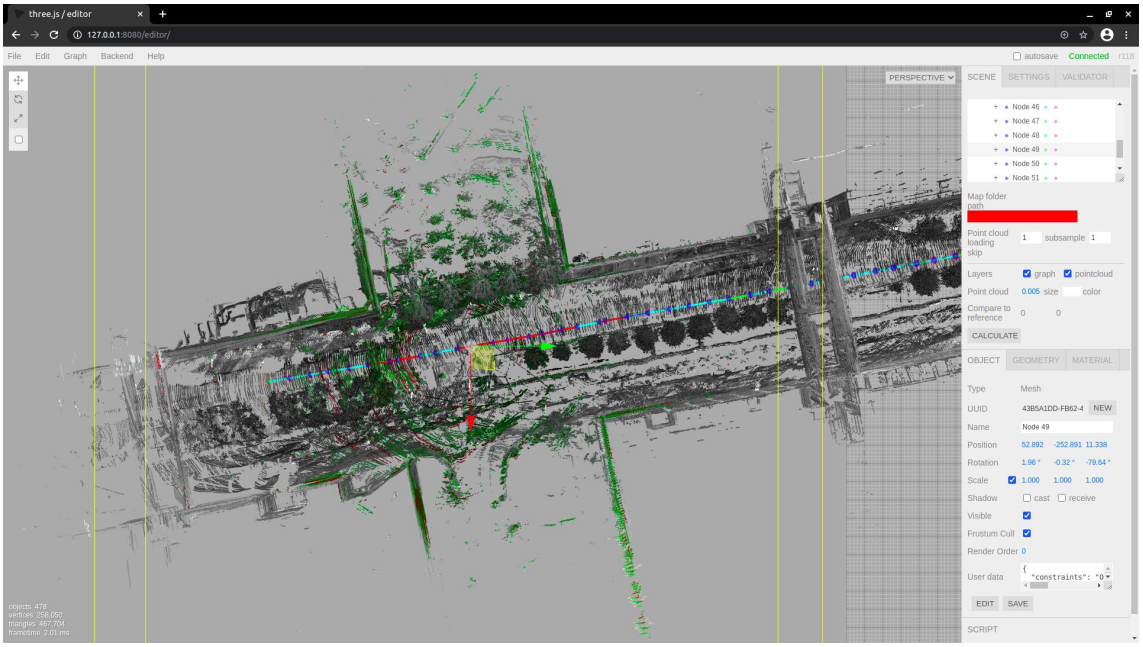

Figure B10: Output map for manipulation sequence OM-9.

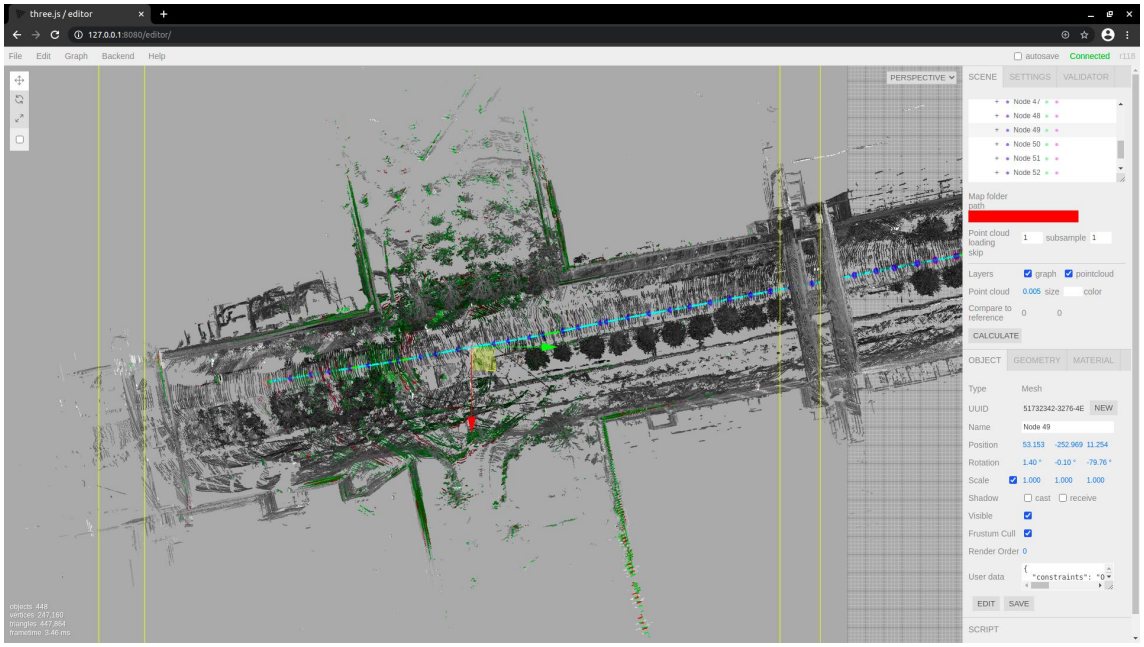

Figure B11: Output map for manipulation sequence OM-10.

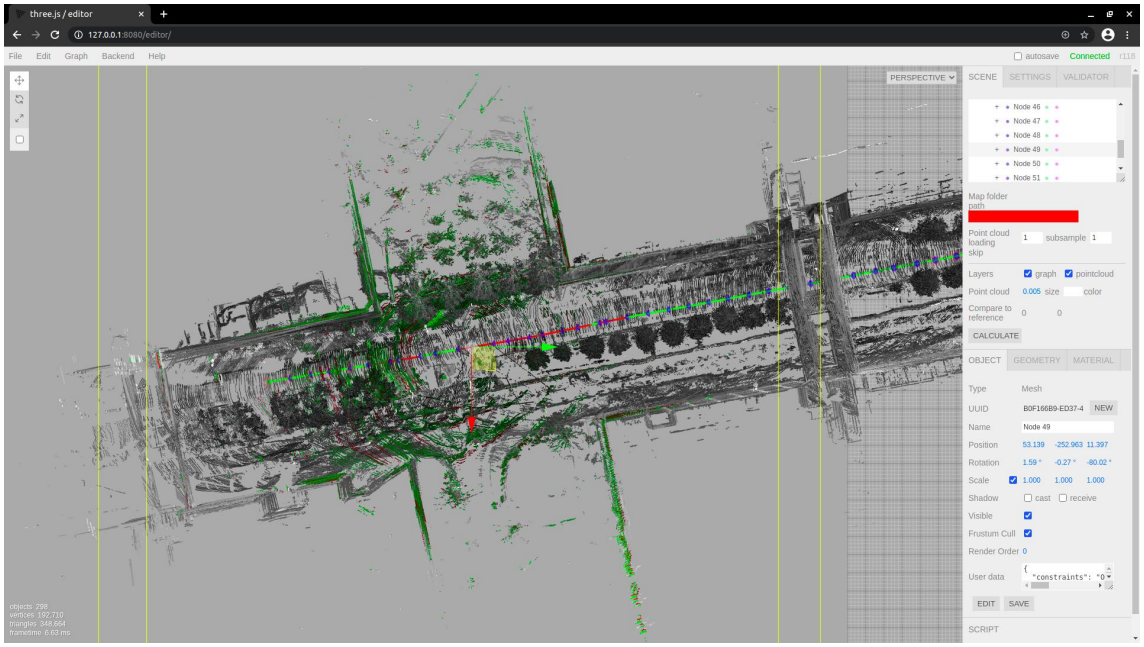

Figure B12: Output map for manipulation sequence OM-11.

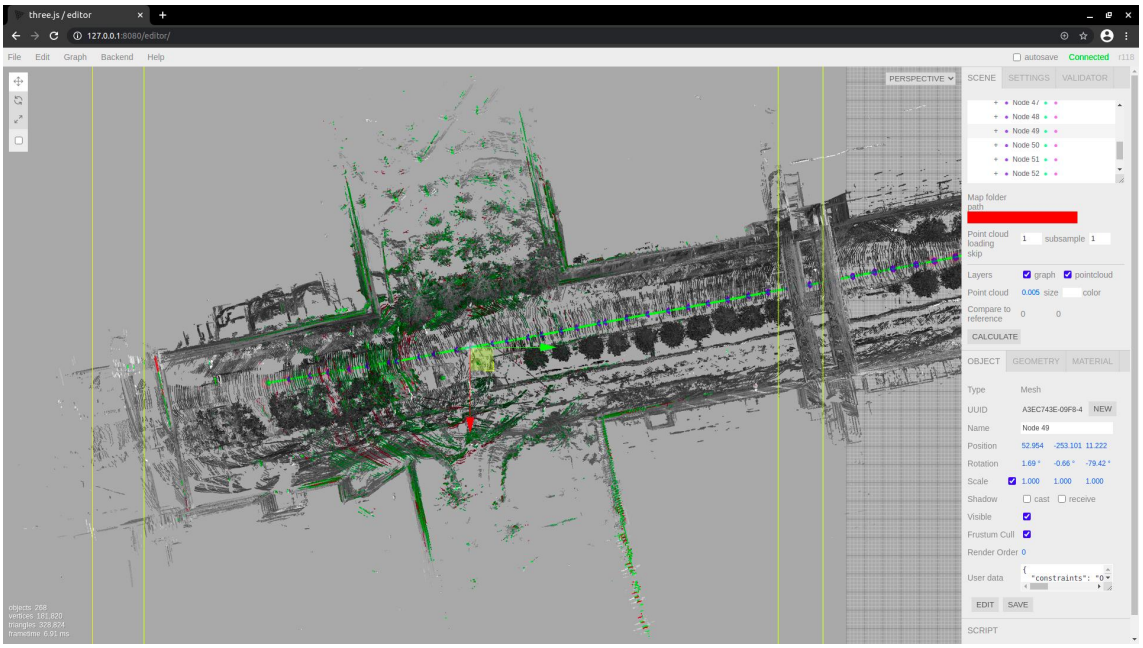

Figure B13: Output map for manipulation sequence OM-12

C Data set 2. Visual inspection results

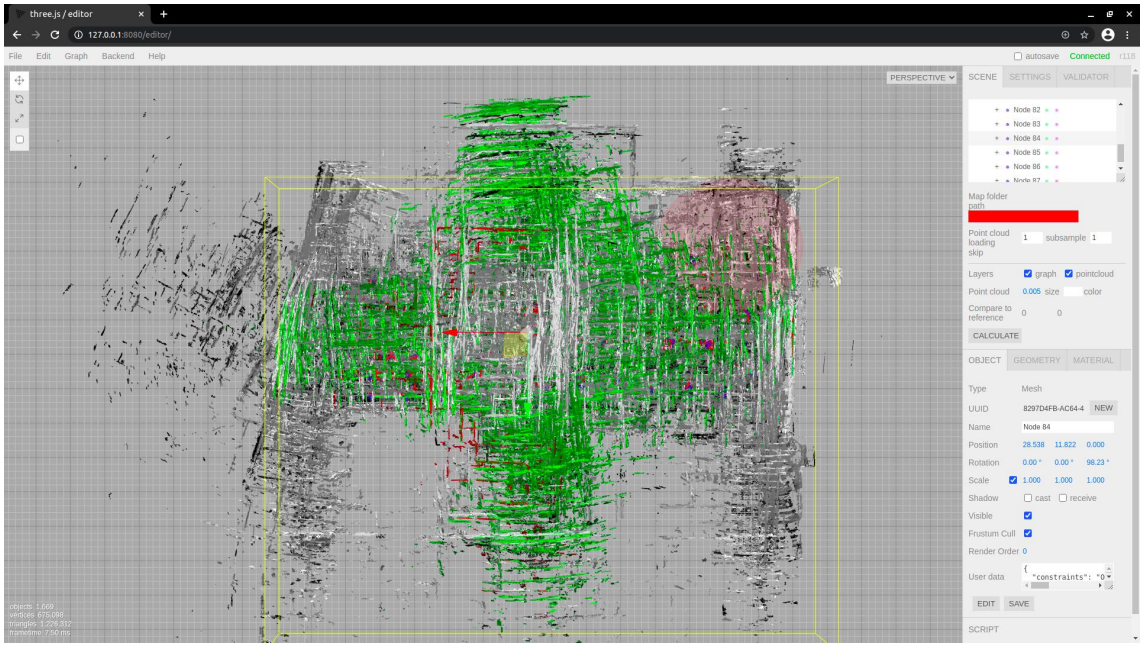

Figure C1: Original distorted map for Data set 2.

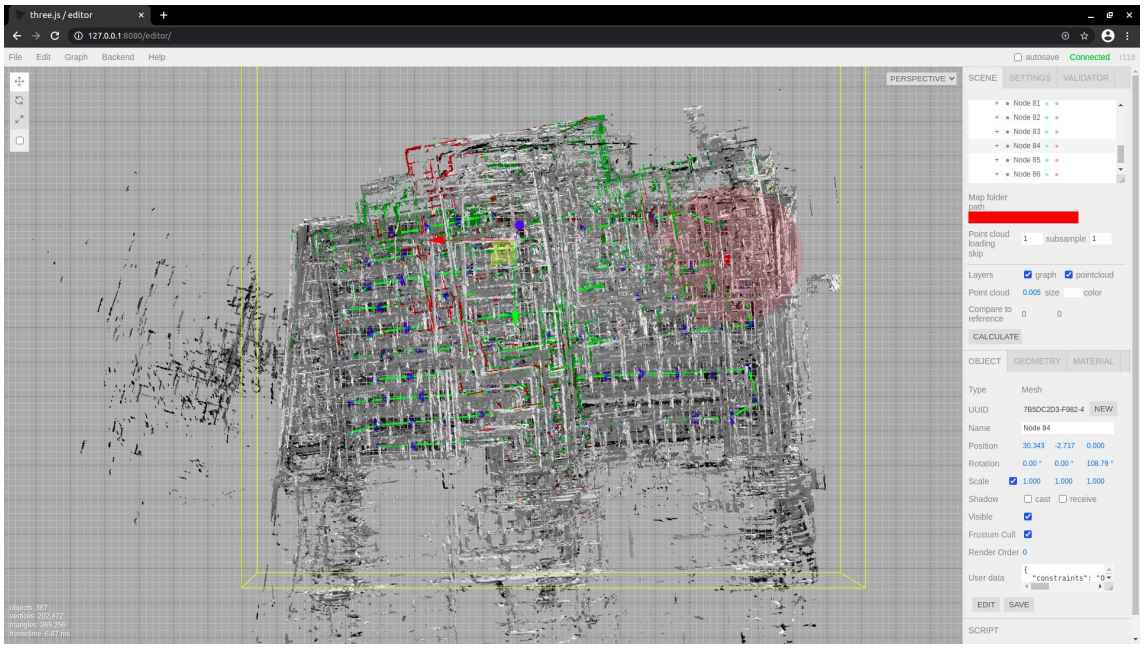

Figure C2: Output map for manipulation sequence IM-1.

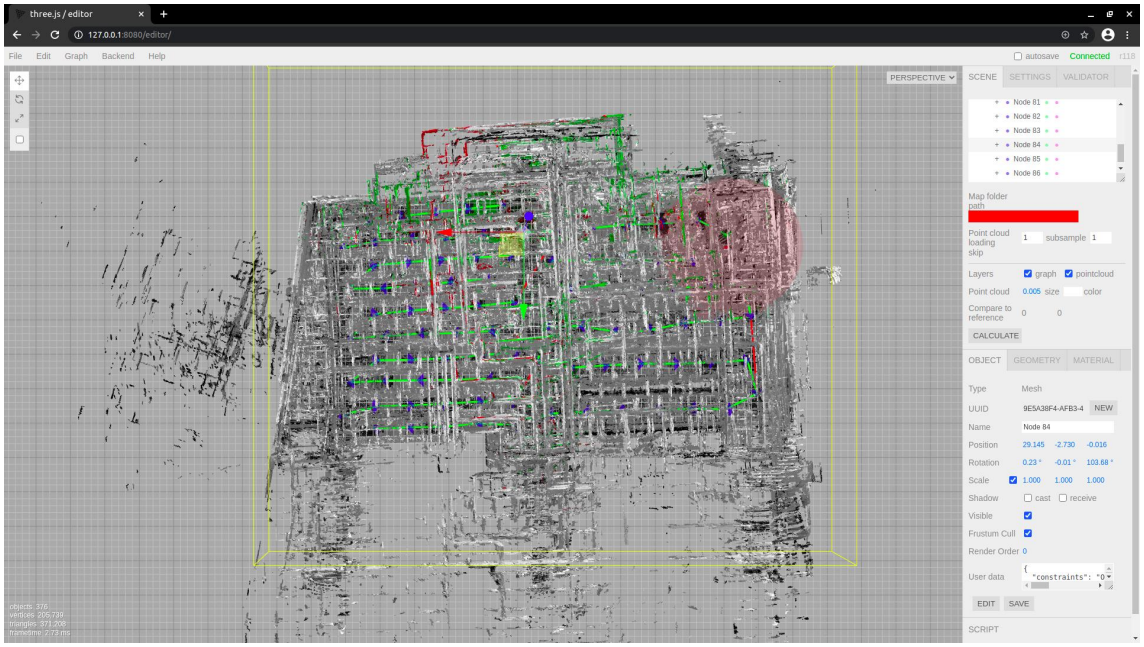

Figure C3: Output map for manipulation sequence IM-2.

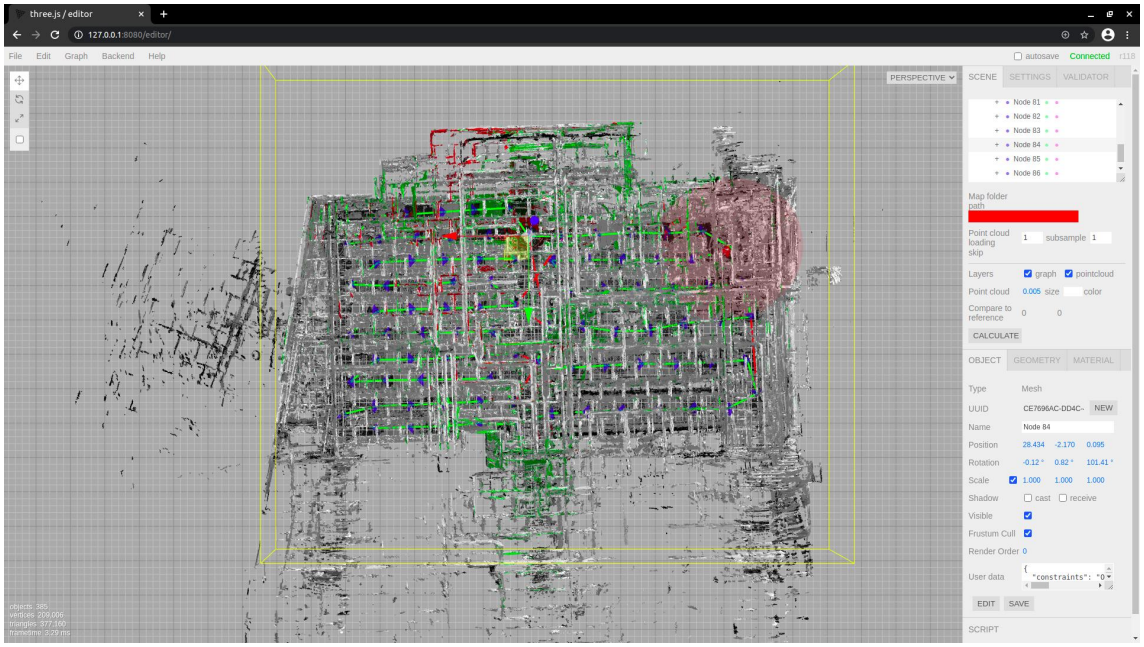

Figure C4: Output map for manipulation sequence IM-3.

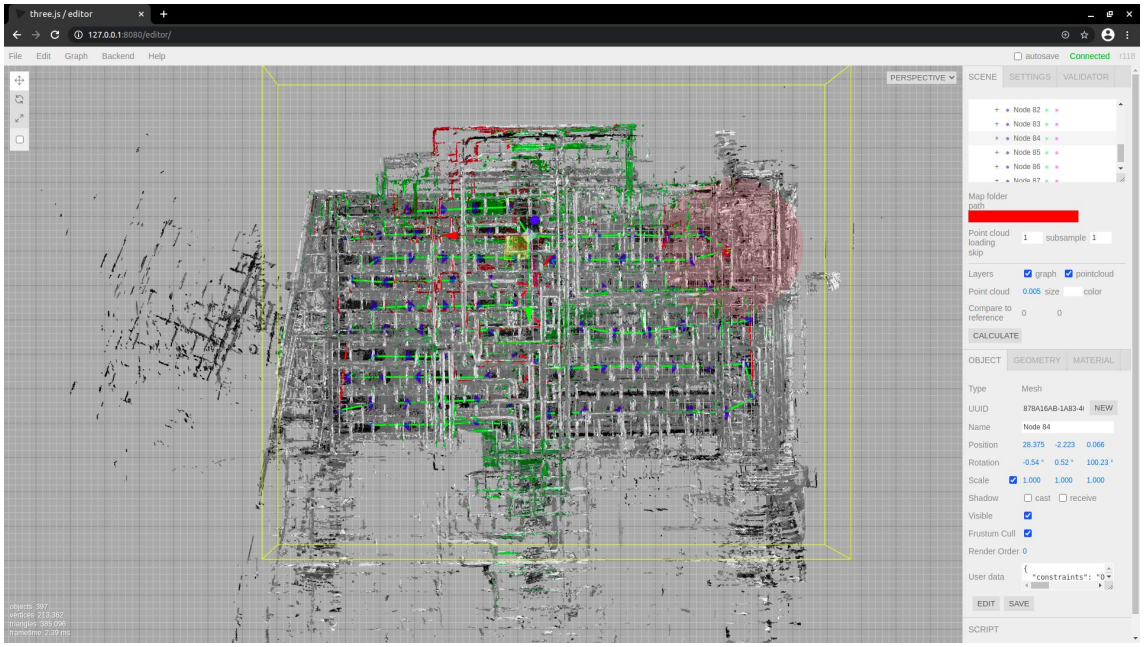

Figure C5: Output map for manipulation sequence IM-4.

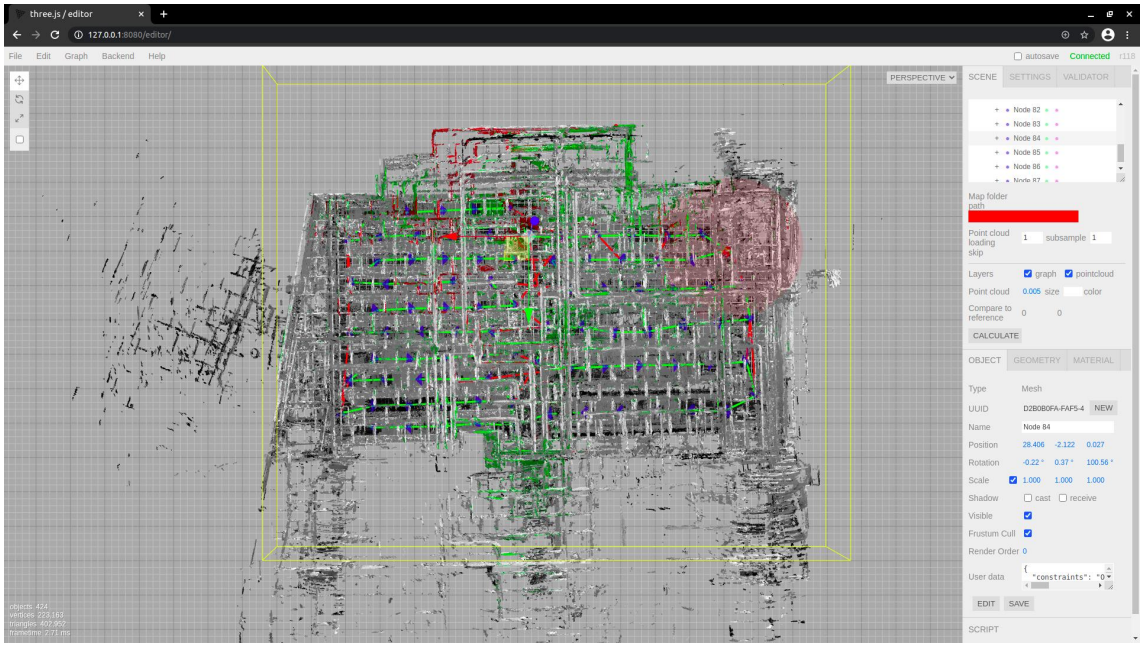

Figure C6: Output map for manipulation sequence IM-5.